1991年10月1日発行（毎月1回1日発行）第10巻10号通巻114号　昭和58年11月2日第三種郵便物認可

# 特集 マシン語との邂逅

IOCS,DOSコールの使い方/実践プログラミング
デバッガで挑戦/避けて通れぬアドレッシングモード

新連載 Creative Computer Music入門

詳報 NEW PrintShop PRO-68K ver.2.0

10
1991

オー／エックス
定価600円

SHARP

仕事だけのパソコンや
ワープロみたいなパソコンは、
いらない。

# 父のパソコンを超えろ。

シャープX68000パソコン教室開催中
- 会場：四谷教室
- コース：入門コース・表集計コース・音楽コース・絵画コース
- 申込受付電話番号(03)3260-8365
- 受講料：2,000円(税別)

夢、創ります。「第1回全日本X68000芸術祭」 作品募集中!

クリエイティブマインドを刺激する全国規模のビッグなオリジナルソフトウェア・作品コンテストです。
ゲーム、ミュージック、グラフィックなどの力作をぜひお寄せ下さい。詳細は店頭でポスター・チラシをご覧ください。

| (〆切り迫る!) | 神奈川 | 9/20(金) | 中部 | 10/4(金) | 北陸 | 10/18(金) | 近畿 | 10/25(金) |
|---|---|---|---|---|---|---|---|---|

(応募・問い合わせ先)■北陸地区〒921石川県石川郡野々市町字御経塚町1096-1シャープエレクトロニクス販売(株)北陸統轄(営)パソコン担当☎0762-49-118(代) ■近畿地区〒556大阪市浪速区恵美須西1-2-9シャープエレクトロニクス販売(株)近畿統轄(営)パソコン担当☎06-631-1181(代)、他地区は右頁をご覧下さい

いまクロック16MHzの俊才、「エクシヴィ」のデビューで5年に及ぶ68000CPUへの探求は、ひとつの結論を得ようとしています。極めたといえば言い過ぎでしょうが、事の深淵に迫ろうと努力するもののみに与えられる深い充足を、私たちスタッフは、これまでX68000を支えていただいたユーザー、ソフトハウス、ハードベンダー諸兄とともに味わいたい心境です。徹底したこだわりと、それを裏付けるアドバンストテクノロジー、世間の逆風を揚力にしてしまう、それなりの魅力と知性を背景として備えたX68000が、パーソナルコンピュータに新しいジャンルを切り拓いてきた歩みは、ご存じの通りです。現在のマルチメディア環境を開発当初から想定していた先見性。一言でいえばクリエイティブマインドということでしょうが、そのグラフィックアビリティ、映像統合コンセプト、サンプリング音源、ウィンドウ環境、そうした単に、とはいえ凄いスペックさえ超えたところにX68000の付加価値は存在します。アプリケーションを走らせるだけのブラックボックス化した、あるいは文房具としてのマシン、それはそれで異論はないのですが、本来的にパーソナルコンピュータがもつ可能性を育む、いわば創造性という観点から物足りなさを覚えることも事実です。X68000は、ある意味ではたいへんな異端児かも知れません。しかし世間から見たその"異能"は、私たちが考えるパーソナルコンピュータとしてはまさにスタンダードに他なりません。いつも新鮮な感動がある、驚きがある。新しい発見がある。"センス"の違いはスペックをも超えて使う人に訴えかける、敢えて68000CPUに執着してきた理由もここにあります。ワークステーションとしての成熟、先見性、創造性の具現化、ユーザーインターフェイスの追求。X68000の進化の過程はここに凝縮されています。

――――――――――――――――新しい「エクシヴィ」がこのコンセプトをどう発展させたか、ご体感ください。

# 瞬速16MHz、エクシヴィ快走。

**16MHzクロック68000搭載**:OSの高速化、アートワークをパワフルにサポートするクロック周波数16MHzの68000CPUを搭載。クリエイティブワークステーションにふさわしいシステムパフォーマンスを実現しました。

**SX-WINDOW ver1.1搭載**:CPUのクロックアップと合わせ、大幅な処理速度の向上を実現。操作性を一段と高めたニューバージョンです。多機能・高速の強力エディタを搭載。文字選択・外字作成ツールも装備して、スムーズな日本語入力環境をサポート。またプリンタデバイスドライバを搭載し、多彩な印字指定が可能です。もちろん、こうした新しい環境がすべてのX68000で享受できることは言うまでもありません。そして待望のウィンドウアプリケーションもリリースされはじめています。

**高密度メモリ拡張環境**:メインメモリは標準で2Mバイト、本体内部のメイン基板上に6Mバイト増設でき、I/Oスロットを使用せず最大8Mバイトの高速メモリアクセスを実現。さらにI/Oスロットへの増設を含め最大12Mバイトまで拡張できます。数値演算プロセッサも本体内に取り付けられます。

※2MB増設メモリ(ボード型)CZ-6BE2A 標準価格59,800円(税別)、2MB増設メモリ(チップ型)CZ-6BE2B 標準価格54,800円(税別)、数値演算プロセッサ(チップ型)CZ-6BP2 標準価格45,800円(税別)を使用。(すべて別売)

●大容量メディア対応、世界標準SCSIインターフェイス標準装備●X68000シリーズとフルコンパチブル設計●高品位なチタンブラックのニューデザインマンハッタンシェイプ●81MバイトSCSI仕様HDD搭載(CZ-644C)/内蔵可能(CZ-634C)●1024×1024ドットの実画面エリアを装備した高解像度表示(最大表示エリア768×512ドット・65,536色中16色表示)、65,536色同時表示(512×512ドット時)、先駆の高解像度自然色グラフィック●AD PCM、ステレオ8オクターブ8重和音FM音源搭載●オートロード・オートイジェクトの1Mバイト5インチFDD2基搭載●マウス・トラックボール標準装備

●写真はCZ-644C-TNとCZ-614D-TN。

本体+キーボード+マウス・トラックボール
CZ-634C-TN(チタンブラック) 標準価格368,000円(税別)
81MB HDタイプ CZ-644C-TN(チタンブラック) 標準価格518,000円(税別)

**SUPER** 本体+キーボード+マウス・トラックボール
CZ-604C-TN(チタンブラック) 標準価格348,000円(税別)
81MB HDタイプCZ-623C-TN(チタンブラック) 標準価格498,000円(税別)

**PRO**II 本体+キーボード+マウス
CZ-653C-BK(ブラック)・-GY(グレー) 標準価格285,000円(税別)
40MB HDタイプCZ-663C-BK(ブラック)・-GY(グレー) 標準価格395,000円(税別)

- 15型カラーディスプレイテレビ(ドットピッチ0.39mm) CZ-605D-BK(ブラック)・-GY(グレー) 標準価格115,000円(スピーカー2個/チルトスタンド同梱・税別)※
- 14型カラーディスプレイテレビ(ドットピッチ0.31mm) CZ-607D-TN(チタンブラック)・-BK(ブラック) 標準価格99,800円(チルトスタンド同梱・税別)***NEW***
- 15型カラーディスプレイテレビ(ドットピッチ0.31mm) CZ-614D-TN(チタンブラック)・-BK(ブラック) 標準価格135,000円(スピーカー2個/チルトスタンド同梱・税別)***NEW***
- 14型カラーディスプレイ(ドットピッチ0.31mm) CZ-604D-BK(ブラック)・-GY(グレー) 標準価格94,800円(スピーカー2個/チルトスタンド同梱・税別)
- 14型カラーディスプレイ(ドットピッチ0.31mm) CZ-606D-TN(チタンブラック)・-BK(ブラック)・-GY(グレー) 標準価格79,800円(チルトスタンド同梱・税別)
- 21型カラーディスプレイ(ドットピッチ0.52mm) CU-21HD(ブラック) 標準価格148,000円(スピーカー2個同梱・税別)

※印の商品は在庫僅少です。

地区予選大会開催中!!お友達を連れてぜひ、ご来場ください。

| 開催日 | 開催地 | 会場 | 応募・問い合わせ先 |
|---|---|---|---|
| 9月22日(日) | 北関東地区 | 護国会館 平安の間<br>宇都宮市陽西町1-37 ☎0286-22-3180 | シャープエレクトロニクス販売(株)<br>北関東統轄(営)パソコン担当 ☎0286-35-1151(代) |
| 10月6日(日) | 神奈川地区 | 神奈川県労働総合センター5F 大講堂<br>横浜市磯子区中原1-1-28 ☎045-773-2250 | シャープエレクトロニクス販売(株)<br>神奈川統轄(営)パソコン担当 ☎045-753-5501(代) |
| 10月20日(日) | 中部地区 | シャープ名古屋ビル 7Fホール<br>名古屋市中川区山王3-5-5 ☎052-323-5111 | シャープエレクトロニクス販売(株)<br>中部統轄(営)パソコン担当 ☎052-323-5111(代) |

●お問い合わせは…
シャープ株式会社
電子機器事業本部システム機器営業部
〒545 大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)
電子機器事業本部液晶映像システム事業部第2商品企画部
〒162 東京都新宿区市谷八幡町8番地☎(03)3260-1161(大代表)

# C O N T

特集 マシン語との邂逅

NEW Print Shop PRO-68K Ver.2.0

ボナンザブラザーズ

ジーザスII

GS規格

(で)のショートプロぱーてぃ




〈スタッフ〉
●編集長／前田 徹 ●副編集長／植木章夫 ●編集／岡崎栄子 浅井研二 山田純二 ●協力／有田隆也 中森 章 林 一樹 荻窪 圭 華門真人 毛内俊行 吉田賢司 影山裕昭 古村 聡 村田敏幸 丹 明彦 三沢和彦 長沢淳博 宮島 靖 金子俊一 浦川博之 石上達也 ●カメラ／杉山和美 ●イラスト／永沢しげる 山田晴久 寺尾響子 ●アートディレクター／島村勝頼 ●レイアウト／元木昌子 AD GREEN ●校正／グループごじら

表紙絵：須藤　牧人

# ENTS

# 1991 OCT. 10

入力から、編集、出力、複写まで、カラーの世界は、シャープです。

## ▶INPUT

X68000用パラレルインターフェイスを標準装備した
高速コンパクト型イメージスキャナ。

**カラーイメージスキャナ JX-220X……標準価格168,000円(税別)**

●A4サイズの原稿を約50秒※1で高速読み取り●CCDセンサー採用。さらに中間調処理でシャープでリアルな画像を再現●ディザパターン指定機能※2や濃度補正機能※2など高度な画像処理機能で緻密な読み取りが可能●解像度200ドット/インチ(約7.9ドット/mm)。ズーム機能で1%きざみの拡大、縮小も可能●色ずれの少ない線順次(1走査)読み取り●X68000シリーズ用「スキャナツール」ソフトを標準装備●プリンタと直接接続することによりダイレクトプリント※3が可能●RS-232Cインターフェイス/X68000シリーズ用専用パラレルインターフェイスを標準装備。

※1：A4、2値出力、コンピュータへの実転送時間。
※2：表記機能はJX-220X本体使用であり、付属ユーティリティ使用時は異なります。
※3：別売のパラレルインターフェイスケーブル(JX-22PC標準価格12,000円(税別)が必要です。

## ▶OUTPUT

3種類の制御コマンドモードを搭載。
質感も鮮やかに再現する高品位カラーイメージジェット。

**カラーイメージジェット IO-735X-B……標準価格248,000円(税別)**

●シャープ独自のIOシリーズコマンド(Gモード)に加え、NM-9900モード(Nモード)、ESC/P24-84C準拠モード(Pモード)をサポート。一般文書の作成から、各種デザイン、建築用パースなどのCAD分野に対応●発色性に優れた普通紙対応の新黒インキ採用。専用紙はもちろんオフィスでよく使われる普通紙にも鮮明カラー印字●プリントバッファメモリ(128KB)の内蔵で、ホストコンピュータの拘束時間を軽減●48ノズル(各色12ノズル)採用の高速印字。A4-1ページを※約90秒でプリント(データ受信時間除く)●ビジネス用途に適したB4横用紙幅対応●OHPフィルム(専用)にも鮮明プリント●ノンインパクト方式ならではの静粛印字●インキ補充は簡単、経済的なカートリッジ方式

※261×174mm領域

**IO-735X-B
対応アプリケーション**

●SX-WINDOW対応ペイントツール
**Easypaint** SX-68K
CZ-263GW 標準価格12,800円(税別)

●WYSIWYGを実現、ドローグラフィックソフト
**CANVAS** PRO-68K
CZ-249GS 標準価格29,800円(税別)

●オリジナリティを活かせるポップアップツール
**NEW Printshop** PRO-68K ver2.0
CZ-221HS 標準価格20,000円(税別)

●マルチワープロ PRO-68K
**Multiword**
CZ-225BS 標準価格32,000円(税別)

●高速カード型リレーショナルデータベース
**CARD** PRO-68K ver2.0
CZ-253BS 標準価格29,800円(税別)

●パソコン通信もできるメモリ常駐型ソフト
**Teleportion** PRO-68K
CZ-258BS 標準価格22,800円(税別)

●これからの高速通信をサポート
**Communication** PRO-68K ver2.0
CZ-257CS 標準価格19,800円(税別)

平成3年9月末日迄
**(バナナキャンペーン実施中!)**
期間中IO-735X-Bを買うと便利なプリントツール(デモ版)がついてくる。

■拡大縮小、マルチ印刷など多彩な印刷機能を装備したプリントツール
BANANA PRINT………………標準価格48,000円(税別)〈発売元：(有)ムーンベース〉☎022(271)9700

資料のご請求・お問い合わせはコンシューマーセンター／東日本OA相談室 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)・西日本OA相談室 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)・名古屋OA相談室 〒454 名古屋市中川区山王三丁目5番5号 ☎(052)332-2611(大代表)・福岡OA相談室 〒816 福岡市博多区井相田2丁目12番1号 ☎(092)575-2381(代表)

SHARP

# 多彩なグラフィック機能搭載
# 多機能ワープロ

マルチワープロ PRO-68K

## Multiword

CZ-225BS　標準価格32,000円(税別)

WYSIWYGを採用したウィンドウモード、エディタ感覚で入力できるテキストモード。さらにクリエイティブマインドを刺激する多彩なグラフィック機能を搭載。X68000のパフォーマンスをフルに活かした、ヒューマンなワープロの誕生です。

- ○最大10文書までの複数文書を同一画面で編集できるウィンドウモード装備。文書間でのカット&ペーストも可能
- ○スピーディな文字入力をサポートするテキストモード
- ○20種類のペン先を使って自由にグラフィックを作成できるグラフィックエディタを装備。タイルパターンは52種類、オリジナルパターンも作成可能
- ○影付文字、袋文字、斜体文字など多彩な文字種、文字間隔もドット単位に指定可能
- ○ビジネス文書に威力を発揮する豊富な改行・罫線機能
- ○用途に合わせて選べる幅広いプリンタサポート。多彩な用紙設定、印刷設定で思い通りのアウトプットが可能
- ○イメージスキャナ入力は、パラレルインターフェイスに対応。ハンディスキャナ入力もサポート。

※メインメモリ2MB必要です。

### ●Multiword講習会のお知らせ

| 日　時 | 場　所 | お問い合わせ先 |
|---|---|---|
| 9月28日(土) | シャープ東京支社 | 03(3260)8365 |
| 9月29日(日) | シャープ東京支社 | 03(3260)8365 |
| 9月29日(日) | 名古屋OAショールーム | 052(323)5150 |
| 10月5日(土) | 福岡OAショールーム | 092(481)2860 |
| 10月9日(水) | 大阪OAショールーム | 06(222)7655 |
| 10月12日(土) | 福岡OAショールーム | 092(481)2860 |
| 10月12日(土) | 横浜OAショールーム | 045(201)6525 |

資料請求券 CZソフト Oh!X 10係

## MONTHLY PICK UP

●シューティングゲーム

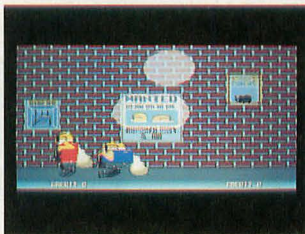
### 中華大仙

CZ-268AS　標準価格7,900円(税別)

©TAITO CORP. 1988

●コミカルアクションゲーム

### ボナンザブラザーズ

CZ-270AS　標準価格9,000円(税別)

©SEGA1990 REPROGRAMMED BY SHARP/SPS
※メインメモリ2MB必要です。

●バイクレーシングゲーム

### ダッシュ野郎

CZ-269AS　標準価格8,800円(税別)

©TOAPLAN Co. Ltd. 1988

●高速カード型リレーショナルデータベース

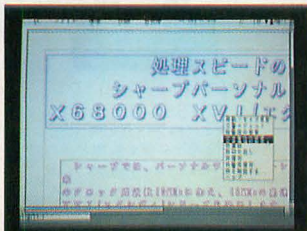
CZ-253BS 標準価格29,800円(税別)

操作性の向上、高速化を図った新マルチウィンドウシステムを搭載したニューバージョンです。一覧表画面入力、グラフ機能などをサポート。キーボード操作にも対応します。

※メインメモリ2MB必要です。
※CARD PRO-68K(CZ-226BS)をお持ちの方には有償バージョンアップを行います。

●各種エディタを装備したレイアウターソフト

### Press Conductor PRO-68K

CZ-266BS　10月発売予定

Zeit社の「書体倶楽部」の全アウトラインフォントに対応。簡単なマウス操作により、机の上で紙片を貼り合わせる感覚で、文章、図形、罫線などをディスプレイ上で自由にレイアウトできます。

※メインメモリ2MB必要です。

●Zeit日本語ベクトルフォントをサポート

CZ-265HS　標準価格20,000円(税別)

効率のよい操作環境を実現。カセットレーベル、カレンダー作成に対応したほか、モノクロデータの編集などグラフィックエディタを強化した高機能テキストエディタを内蔵しています。

※メインメモリ2MB必要です。
※NEW Print Shop PRO-68K(CZ-221HS)をお持ちの方には有償バージョンアップを行います。

●SX-WINDOW対応ペイントツール

### Easypaint SX-68K

CZ-263GW　標準価格12,800円(税別)

マウスによる簡単操作、65,536色中16色の多彩なカラー表現、SX-WINDOW対応初のペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間でデータのやりとりもOK。

※メインメモリ2MBおよびSX-WINDOW ver.1.1が必要です。

《お詫びと訂正》■弊社発行のX68000ソフト情報誌「ソフトウェアフィールド」20号において、一部標準価格に誤りがありますので訂正させていただくとともに、謹んでお詫び申し上げます。

- Musicstudio PRO-68K ver2.0(CZ-261MS)……(誤)標準価格 18,800円(税別)→(正)標準価格 28,800円(税別)
- 中華大仙(CZ-268AS)……(誤)標準価格 8,800円(税別)→(正)標準価格 7,900円(税別)
- 光磁気ディスクユニット(CZ-OM01)……(誤)標準価格 29,800円(税別)→(正)標準価格450,000円(税別)
- SCSIボード(CZ-6BS1)……(誤)標準価格450,000円(税別)→(正)標準価格 29,800円(税別)

※CZ-253BS、CZ-265HSの有償バージョンアップについては、下記にお問い合わせください。

●お問い合わせは…シャープ㈱電子機器事業本部液晶映像システム事業部第2商品企画部 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)へ。シャープ株式会社

お望みのパワーシステムへ。

PROⅡ

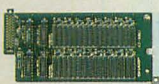
## ボード

### 拡張メモリ

NEW

2MB増設RAMボード
(CZ-634C/644C専用)
**CZ-6BE2A**
標準価格59,800円(税別)
※2MB増設RAM(CZ-6BE2B)専用ソケットを2個用意しています。

NEW

2MB増設RAM
(CZ-634C/644C専用)
**CZ-6BE2B**
標準価格54,800円(税別)
※本増設RAM(CZ-6BE2B)は、2MB増設RAMボードが必要です。CZ-6BE2A上の専用ソケット(2個用意)に装着ください。
※取付に関してはシャープお客様ご相談窓口にてご相談ください。

1MB増設RAMボード
(CZ-600C専用)
★**CZ-6BE1**
標準価格35,000円(税別)

1MB増設RAMボード
(CZ-601C/611C/652C/653C/662C/663C用)
**CZ-6BE1B**
標準価格28,000円(税別)

2MB増設RAMボード※6
**CZ-6BE2**
標準価格79,800円(税別)

4MB増設RAMボード※6
**CZ-6BE4**
標準価格138,000円(税別)

### インターフェイス

SCSIボード※7
**CZ-6BS1**
標準価格29,800円(税別)
(ソフトウェア(SCSIユーティリティ)同梱)

ユニバーサルI/Oボード
★**CZ-6BU1**
標準価格39,800円(税別)

GP-IBボード
★**CZ-6BG1**
標準価格59,800円(税別)

増設用RS-232Cボード
(2チャンネル)
★**CZ-6BF1**
標準価格49,800円(税別)

### MIDI

MIDIボード
★**CZ-6BM1**
標準価格26,800円(税別)

### FAX

FAXボード
**CZ-6BC1**
標準価格79,800円(税別)

### 数値演算プロセッサ

数値演算プロセッサボード
**CZ-6BP1**
標準価格79,800円(税別)

NEW

数値演算プロセッサ
(CZ-634C/644C専用)
**CZ-6BP2**
標準価格45,800円(税別)
※取付に関してはシャープお客様ご相談窓口にてご相談ください。
※特別ケース入りです。

## ネットワーク

### モデム

モデムユニット※8
**CZ-8TM2**
標準価格49,800円(税別)
(RS-232Cケーブル同梱)

### RS-232Cケーブル

RS-232Cケーブル
(平行接続型)
**CZ-8LM1**
標準価格7,200円(税別)

RS-232Cケーブル
(クロス接続型)
**CZ-8LM2**
標準価格7,200円(税別)

### LANボード

LANボード
**CZ-6BL1**
標準価格268,000円(税別)
(イーサネット用)

**CZ-6BL2**
標準価格298,000円(税別)
(イーサネット/チーパネット両用)
※電源ユニット・ソフトウェア
(ネットワークドライバVer1.0)同梱

## 入力

インテリジェントコントローラ
**CZ-8NJ2**
標準価格23,800円(税別)

マウス・トラックボール
**CZ-8NM3**
標準価格9,800円(税別)

トラックボール
**CZ-8NT1**
標準価格13,800円(税別)

マウス
**CZ-8NM2A**
標準価格6,800円(税別)

ジョイカード
**CZ-8NJ1**
標準価格1,700円(税別)

## その他

### 拡張スロット

拡張I/Oボックス(4スロット)
(CZ-600C/601C/602C/603C/604C/611C/612C/613C/623C/634C/644C用)
**CZ-6EB1-BK**
★**CZ-6EB1**
標準価格88,000円(税別)

### スピーカー

アンプ内蔵
スピーカーシステム(2本1組)
**AN-S100**
標準価格36,600円(税別)

### システムラック

システムラック
(CZ-600C/601C/602C/603C/604C/611C/612C/613C/623C/634C/644C用)
**CZ-6SD1**
標準価格44,800円(税別)

■本広告に掲載しております拡張ボード類のうち、CZ-634C/644Cの16MHzモードで動作しないものが一部あります。 ★印の商品は在庫僅少です。
■製品改良のため仕様の一部を予告なく変更することがあります。またこの広告の色調は印刷のため実物とは多少異なる場合もありますのであらかじめご了承ください。

CZ-600C用)、CZ-6BE1B 標準価格28,000円(税別・CZ-601C、CZ-611C、652C、653C、662C、663C用)を増設してください。 ※7 CZ-600C、601C、602C、603C、611C、612C、613Cに装着の場合、I/Oスロット2に装着ください。CZ-652C、653C、662C、663Cに装着の場合はI/Oスロット4に装着ください。また、CZ-6BG1、6BU1、6BL1、6BL2、6BN1などのボードは、接続コネクタとの関係で本ボードとの併用はできませんのでご注意ください。なお、本ボードはX68000用OS Human 68K ver.2.0以上にてご使用ください。 ※8 モデムユニットCZ-8TM2に同梱のソフトはX1/X1ターボシリーズ用です。

子機器事業本部液晶映像システム事業部第2商品企画部 〒162 東京都新宿区市谷八幡町8番地 ☎(03)3260-1161(大代表)

X68000

# 絶賛冒険中!!

# ロードス島戦記 ©
## 灰色の魔女

原作:安田　均/水野　良
キャラクターデザイン:出渕　裕

■標準価格に消費税は含まれておりません。お買い上げの際に別途消費税をお支払い下さい。
■通信販売ご希望の方は、住所、氏名、電話番号、商品名、機種名、メディアを明記の上、現金書留または郵便振替(大阪8-303340)にてお申し込み下さい。送料は無料ですが、標準価格に消費税の3%を加えた金額をお送り下さい。

Humming Bird Soft™
株式会社エム・エー・シー ハミングバードソフト
〒530 大阪市北区曽根崎2丁目2番15号 TEL 06(315)8255

開戦迫る!

名作シューティング

ひしょうざめ

**10月25日**
**発売予定**

*X68000用　予価¥8,800(税別)*

**革命的プロジェクト進行中!!**

**次はキミが創る!!** KANEKOの次期作品の開発スタッフを募集します。X68000版「究極タイガー」、「達人」、「鮫・鮫・鮫」をキミの手で創って下さい。このプロジェクトに関する詳細は☎03-5261-2147 鵜藤まで。

**X68000名作シューティング・シリーズ**

| | |
|---|---|
| 究極タイガー | '92年春発売予定 |
| 達人 | '92年夏発売予定 |
| 鮫・鮫・鮫 | '92年秋発売予定 |

**プレゼントキャンペーン実施** X68000名作シューティング・シリーズ全4作お買い上げの方に、4タイトルが収納できる、超豪華オリジナル・パッケージをもれなくプレゼント!

KANEKO®

株式会社　金子製作所
〒177　東京都練馬区石神井台8丁目23番21号
TEL.03(3921)9661

信頼と安さの TSUKUMO

ソフト・ハード なんでもツクモ

掲載商品2万円以上送料無料(離島を除く)

# 情報キャッチステーションツクモ

新製品の情報もいろいろな機種の情報も、ツクモに行けばあなたのもの。歩きまわる必要はありません。ツクモだけで十分です!!

★★★★★★★★★★シャープ製品は、小さいモノはポケコンから大きいモノは液晶ビジョンまで、何でも揃う!★★★★★★★★★★

お気軽にお越し下さい。親切丁寧に対応させていただきます。他じゃマネできません。
ツクモだからちゃんと対応できるんです。

## やっぱりXVI これが一番!

### X68000 XVI 快速16MHz

- CPUクロック周波数スピードアップ(16MHz)
- 増設メモリ本体内蔵可能(8MBまで)
- NEW SX-WINDOW搭載

■X68000XVI(CZ-634C-TN)
標準タイプ……定価¥368,000
■X68000XVI-HD(CZ-644C-TN)
HD内蔵タイプ……定価¥518,000

(買い換え・下取りも取り扱っております。是非、お尋ね下さい。)

## ツクモX68000用TSドライブ

大好評につき在庫お問合せ下さい。

「◀-----▶目のつけどころがツクモでしょ。」

X68000シリーズ用3.5インチフロッピーディスクドライブ **TS-3XR1** 定価¥44,800

仕様
●1ドライブタイプ●3.5インチ2DD/2HD対応ドライブ使用●2DD用ディバイスドライバ付属

ツクモ特価¥35,800

※SX-WINDOW/OS-9は対応しておりません。

## 今月の目玉品!!

AIWAファクシミリ・アダプター
**FILEFAX FF-P4800** 定価¥98,000
4800/2400(自動フォールバック)300bps、GIIIファクシミリ送信可。付属ソフトで受信も可。ワンタッチダイヤル10ヵ所、リダイヤル可。(ソフトはPC-98用です。)

69% off 限定特価¥29,800

## X68000用増設メモリーボード

1MB増設RAMボード (ACE/PRO/PRO2シリーズ用) 特価¥17,500
2MB増設RAMボード‥特価¥34,800
4MB増設RAMボード‥特価¥61,500
※計測技研のメモリーボードも取扱っておりますので、価格についてはお尋ね下さい。

買い取りのツクモニューセンター店

## ツクモ買い取りセンター

―好評買い取り中―

電話受付(03)3251-9977(AM11:00~PM5:00迄)
FAX受付(03)3251-0299(24時間)

## X68000用ハードディスク

―大容量記憶装置―

80MB SCSI/SASI両対応タイプ
**TX-80** 定価¥108,000
特価¥88,000(消費税別途¥2,640)

130MB SCSI対応タイプ
**TX-130** 定価¥138,000
特価¥110,000(消費税別途¥3,300)

180MB SCSI対応タイプ
**TX-180**
定価¥185,000
特価¥148,000
(消費税別途¥4,440)
SCSIタイプHDDの場合、本体がSUPER/XVI以外の場合にはSCSIボード(CZ-6BS1)が必要です。

## ツクモグローバルカード

大人気/入会者募集中/
国内・外で活躍/使って便利、持ってて安心/ツクモグローバルカードはジャックス・VISAとの提携カードです。ツクモ各店でのお買物がらくらくできるうえに、国内はもとより海外での分割ショッピングもOK/
お申し込みは☎03(3251)9898又は店頭にて/

18才以上なら学生でもOK。

※各店頭では、JCB・日本信販・DC・セントラル・マスター他各種カードも取扱っております。

## 更に大容量が欲しい方は…

**SONY光磁気ディスクユニットセット**
NWP-539N(光磁気ディスクドライブ)
SCSIケーブル
光磁気カートリッジ
SCSIインターフェースボード

ツクモ特価¥398,000
シャープ純正「CZ-6MO1」も特価販売中/

## 開発ツール

■C Compiler PRO-68K Ver2.0 定価¥~~44,800~~
■XBAS TO C CHECKER PRO-68K 定価¥~~9,800~~

## パソコン通信

■た~みのる 2……ツクモ特価¥14,200
■一流メーカー2400ボーMNP5対応モデム ツクモ特価¥25,800
■一流メーカー2400ボーMNP5&V42bis対応モデム……ツクモ特価¥31,800

## 電子手帳

■ハイパー電子システム手帳
PA-9500……定価¥48,000 ツクモ特価¥43,000
PA-9550……定価¥~~59,000~~
■スタイリッシュ電子システム手帳
PA-X1・定価¥29,800 ツクモ特価¥26,000
■Teleportion PRO68K……定価¥~~22,800~~
■CE-300L電子手帳通信ケーブル……ツクモ特価¥2,380

※ポケットコンピュータも取扱っております。価格はお尋ね下さい。

## ビジネスツール

■Hyper WORD……定価¥~~39,800~~
■Multiword NEW……定価¥~~32,000~~
■FIXER Ver4.0……ツクモ特価¥15,800
■CARD PRO-68K Ver2.0 NEW 定価¥~~29,800~~

## アートツール(ハード)

■JX-220X A4サイズカラーイメージスキャナー……定価¥~~168,000~~
■HGS-68 ファインスキャナーX68……ツクモ特価¥31,800
■CZ-6VT1 カラーイメージユニット 定価¥~~69,800~~
■CZ-6BV1 ビデオボード……定価¥~~21,000~~
■XAV-1SアナログRGB・S端子変換ユニット ツクモ特価¥6,890
■CZ-8PC5 48ドットカラー漢字熱転写プリンター……定価¥~~96,800~~

## アートツール(ソフト)

■CANVAS PRO-68K……定価¥~~29,800~~
■Easy Paint SX-68K(CZ-263GW) 定価¥~~12,800~~
■Z's STAFF PRO-68K Ver2‥ツクモ特価¥46,400
■マジックパレット……ツクモ特価¥15,800

冬のボーナス一括払い受付中!! 詳しくは☎03(3251)9911へ

## X68000 ユーザー必須のコンピュータミュージック特別セット

**Aセット**
- CM-32L……¥69,000
- SX-68M-II……¥21,000
- Musicstudio Mu-1 Ver1.4……¥19,800

合計定価¥109,800
ツクモ特価¥88,000 (消費税別途¥2,640)
クレジット例(18回払・税込) 初回¥7,223+月々¥5,600×17回

**Bセット**
- CM-64……¥129,000
- SX-68M-II……¥21,000
- Musicstudio Mu-1 Ver1.4……¥19,800

合計定価¥169,800
ツクモ特価¥138,000 (消費税別途¥4,140)
クレジット例(24回払・税込) 初回¥7,603+月々¥6,900×23回

**マニアセット** NEWセット
- SC-55(ローランドサウンドキャンバス)……¥69,000
- SX-68M-II……¥21,000
- Mu-1 SUPER……¥39,800

合計定価¥129,800
ツクモ特価¥99,000 (消費税別途¥2,970)
クレジット例(10回払・税込) 初回¥11,517+月々¥10,900×9回

**SUPERマニアセット** NEWセット
- CM-64……¥129,000
- SX-68M-II……¥21,000
- Mu-1 SUPER……¥39,800

合計定価¥189,800
ツクモ特価¥154,000 (消費税別途¥4,620)
クレジット例(18回払・税込) 初回¥10,940+月々¥9,900×17回

Aセット と Bセット の場合、「Musicstudio PRO68K Ver2.0」又は、「Music PRO68K <MIDI>」のソフトの場合には¥9,500プラスになります。また、これらのソフトウェアがバージョンアップにより価格が変更になった場合には変更となります。

**ローランド 追加オプション機器**
ステレオマイクロモニター CS-10……定価¥~~17,000~~
MIDIキーボードコントローラー PC-200……定価¥~~36,000~~
はなうたくん CP-40……定価¥~~33,000~~

7号店2F(MIDI専門フロア)にもございます。☎03(3253)4199担当・佐々木

商品のご注文は通販受注専用センター フリーダイヤル 0120-377-999
商品についての詳しいお問い合わせは各店、又は☎03(3251)9911へ



ツクモは「スーパーX PRO SHOP」です。

PRO STAFF **ツクモ**

パソコン本店 荒井
N・C店 福地

九十九電機(株)
〒101-91 東京都千代田区神田郵便局私書箱135号
★商品のご注文は在庫確認の上お願いします。★表示価格には消費税は含まれておりません。

ツクモパソコン本店2F ☎03-3253-5599(担当/荒井) 休毎週木曜(10/10を除く)

便利で安心な通信販売
ツクモ通販センター☎03-3251-9911

■ツクモニューセンター店 ☎03-3251-0987(担当/福地) 休毎週木曜(10/10を除く)
■ツクモ5号店 ☎03-3251-0531(担当/森) 休毎週木曜(10/10を除く)
■ツクモAV/カメラ館B1 ☎03-3254-3999(担当/川名) 休毎週水曜
■名古屋1号店 ☎052-263-1655(担当/吉高) 休毎週月曜(9/23を除く)
■名古屋2号店 ☎052-251-3399(担当/横山) 休毎週水曜
■ツクモ札幌店 ☎011-241-2299(担当/田口) 休毎週木曜(10/10を除く)

| カード払い | 全国代金引き換え配達 | クレジット払い | 現金書留払い | 銀行振込払い | 各種リース払い |
|---|---|---|---|---|---|
| 通信販売での御利用カード、ツクモグローバルカード、VIPカード、セントラル、ジャックス※御本人様より電話で通信販売部へお申し込み下さい。 | お申し込みは☎03-3251-9911へお電話1本/配達日の指定もできます。 | 月々¥3,000以上の均等払いも頭金なし、夏・冬ボーナス2回払いも受付中/ | 〒101-91東京都千代田区神田郵便局私書箱135号 ツクモ通販センター Oh!X係 | 事前に☎でお届け先をご連絡下さい。三和銀行 秋葉原支店(普)1009939 ツクモデンキ | くわしくは各店にお問い合せ下さい。ケースに合わせてご相談にのります/ |

◆◆◆◆企業の方へ…お見積りはFAXで。ツクモパソコン本店FAX03-3253-5199担当/荒井へ◆◆◆◆

映像文化の歴史を辿る(たど)と、映画もテレビも初期の時代には先輩達のいない職場で、若い人達が伸び伸びと思う存分力を出し、素晴らしい傑作を生みました。

いま、TVゲーム界がその時期に当たるでしょう。そのチャンスを生かしてプロの道をと思っているキミに、ゲーム・ソフトの制作技術を教える学校がある、それがヒューマン・クリエイティブ・スクールです。　パソコンの経験がなくても、一人一人に適応した環境でクラス編成して、基本から学習するので安心できる。その上、1人2台のコンピュータが使える。講師陣は第一線で活躍している現役。　このようにゲーム業界での活躍に夢を託す人のために、充分お応えする環境を整えてHCSは情熱的、個性的、生徒を募集しています。

**■募集定員**

☆コンピュータ・ゲーム課程(2年制)…200名
アセンブリ・プログラミング、CG及びゲーム・プランニング、コンピュータ・サウンド、ゲーム・制作実習ほか。

☆ニューメディア・プロデュース&CG課程(1年制)…100名
プロデューシング、アニメーション技術、コンピュータ基礎、コンピュータ・ゲーム基礎、コンピュータ・グラフィックス、

**■学校説明会**

10月13日(日)・11月10日(日)・12月8日(日)
※詳しくは、お電話でお問い合わせ下さい。

# 全国通販

■アフターサービス万全のサポート体制
●下取・買取は電話で見積りしております。責任を持って下取りさせて頂きます。

SHARP 認定 PPO-SHOP **O.A.ランド**

(TEL) 03-3770-8855

| 営業時間 | |
|---|---|
| 平日 | AM10:00〜PM7:00 |
| 土日・祭日 | AM10:00〜PM6:00 |

▶9・18〜10・17

SHARPのことならなんでもおまかせ!!

大徳買セール!安く値切ってネ。(本体セット:送料・消費税込み)
お電話下さい。㊙価格をお知らせいたします。

流通事情により、広告表示価格は、お安くなる場合がありますので、ドンドンお電話下さい。

CYBER STICK
■CZ-8NJ2 (定価￥23,800) OAランド特価 ▶￥18,000

**電子手帳** ●見やすい漢字4桁表示!! 情報化時代の必需品!!
■PA-9500(￥48,000)…▶特価￥38,000
■PA-8500(￥28,000)…▶特価￥15,000
■PA-7500(￥22,000)…▶特価￥12,000

## SHARP X68000シリーズセット(送料・消費税込み)

### X68000XVI

①CZ-634-TN+CZ-614D-TN
定価合計￥503,000

| 回数 | 月々 |
|---|---|
| 12回 | ￥33,100 |
| 24回 | ￥17,600 |
| 36回 | ￥12,200 |
| 48回 | ￥9,600 |

②CZ-634C-TN+CZ-607D-TN
定価合計￥467,800

| 回数 | 月々 |
|---|---|
| 12回 | ￥30,800 |
| 24回 | ￥16,300 |
| 36回 | ￥11,400 |
| 48回 | ￥8,900 |

③CZ-634C-TN+CZ-606D-TN
定価合計￥447,800

| 回数 | 月々 |
|---|---|
| 12回 | ￥29,500 |
| 24回 | ￥15,700 |
| 36回 | ￥10,900 |
| 48回 | ￥8,500 |

■CZ-634C■ 特価 ￥TEL下さい!!

■CZ-644C■ 特価 ￥TEL下さい!!

### X68000XVI-HD

①CZ-644C-TN+CZ-614D-TN
定価合計￥653,000

| 回数 | 月々 |
|---|---|
| 12回 | ￥42,800 |
| 24回 | ￥22,700 |
| 36回 | ￥15,800 |
| 48回 | ￥12,400 |

②CZ-644C-TN+CZ-607D-TN
定価合計￥618,700

| 回数 | 月々 |
|---|---|
| 12回 | ￥40,600 |
| 24回 | ￥21,500 |
| 36回 | ￥15,000 |
| 48回 | ￥11,700 |

③CZ-644C-TN+CZ-606D-TN
定価合計￥597,800

| 回数 | 月々 |
|---|---|
| 12回 | ￥39,300 |
| 24回 | ￥20,800 |
| 36回 | ￥14,500 |
| 48回 | ￥11,400 |

XVIお買い上げの方に❶ニュージーランドストーリー ❷V-BALL ❸ジョイカード(連射式) ❹ディスケット20枚プレゼントいたします!!
現金でお買い上げの方には、さらに超特値でお出ししてます。ぜひ一度TEL下さい!!

上記組合せのディスプレイ(モニター)変更自由!!
詳しくは、お電話にてお問い合せ下さい!!

## 周辺機器コーナー 電話で値切ろう。

### プリンターセットコーナー

①CZ-8PC5 NEW 定価￥96,800
●48ドット ●熱転写カラー 漢字プリンター
大特価TEL下さい!!

②CZ-8PK10(24ピン漢字プリンター136桁)
定価￥97,800…特価￥71,000

③CZ-8PG1(24ピンカラー漢字プリンター80桁)
定価￥130,000…特価￥93,000

④CZ-8PG2(24ピンカラー漢字プリンター136桁)
定価￥160,000…特価￥114,000

OAランド特選品!!

■IO-735XB(定価￥248,000)
●カラーイメージジェットプリンター ケーブル付 特価￥169,000

### X68000用ハードディスク

■SCSIタイプ TOWNSでもOK
●アイテック
①TX-80S(￥108,000)…
②TX-130S(￥138,000)…
③TX-180S(￥185,000)…
特価価格!! くわしくはTELで下さい!! さらに値引します。

■SASIタイプ
●ロジテック
①SHD-40(￥99,800)…特価￥60,000

※X68000SUPER/XVI以外の機種では、SCSIボードが必要となります。

★SCSIボード…特価￥22,000
★光ディスク…特価￥320,000
★JX220X…特価￥TEL下さい!!

### X68000用周辺機器コーナー

①CZ-6VT1(カラーイメージユニット)
定価￥69,800…特価￥51,500
②CZ-8NS1(カラーイメージスキャナー)
定価￥188,000…特価￥135,000
③CZ-6BM1(MIDIボード)
定価￥26,800…特価￥20,000
④CZ-6BE2A(2MB増設RAMボード)
定価￥59,800…特価￥44,000
⑤CZ-6BE2B(2MB増設RAM)
定価￥54,800…特価￥40,500
⑥CZ-6BP2(数値演算プロセッサ)
定価￥45,800…特価￥33,800
⑦CZ-6EB1(拡張I/Oボックス=4スロット)
定価￥88,000…特価￥65,000
⑧CZ-6BP1(数値演算プロセッサボード)
定価￥79,800…特価￥59,000

■ゲームソフト、ビジネスソフト2割、3割引きは、当り前で〜す。お問い合せ下さい。

## ★FUJITSU FM TOWNSシリーズ

①FM-TOWNS20F
FM-TOWNS20F
FMT-DP533
FMT-KB105
定価￥422,800
特価￥255,000

②FM-TOWNS40H
FM-TOWNS40H
FMT-DP533
FMT-KB105
定価￥573,100
特価￥375,000

③FM-TOWNS80H
FM-TOWNS80H
FMT-DP533
FMT-KB105
定価￥772,800
特価￥473,000

■プリンター
●CITY RITER (PR-40T)
定価￥120,000…￥TEL下さい!!
●FMPR-204B
定価￥80,000…特価￥43,000

■RAMボード
●ハ戸●HM-02T 2M
定価￥59,800…￥TEL下さい!!
●IOデータ●FJ-SIM32-2M
定価￥27,000…特価￥20,500
●メルコ●XMT2000 2M
定価￥28,000…特価￥20,500

### 《計測技研》増設メモリ&プロセッサ

●高速増設メモリと数値演算プロセッサが一つのボードになった!!●

●KGB-X68PRKII-02(￥55,000)…特価￥42,800
● PRKII-04(￥90,000)…特価￥70,200
● PRKII-06(￥125,000)…特価￥97,500
● PRKII-08(￥160,000)…特価￥124,800
● PRKII-12(￥85,000)…特価￥66,300
●KGB-X68PRKII-14(￥120,000)…特価￥93,600
● PRKII-16(￥155,000)…特価￥121,000
● PRKII-18(￥190,000)…特価￥148,000
●MC-6888 IRC(￥38,000)…特価￥28,500

### I・Oデータ増設RAMボード

■PIO-6BE1-A (1MB) 定価￥25,000 特価￥17,300
■PIO-6BE2-2M (2MB) 定価￥50,000 特価￥33,500
■PIO-6BE4-4M (4MB) 定価￥88,000 特価￥58,500

### ★OAランド推奨ソフト

Ⓐ Easy Paint SX 68K (CZ-263GW) 特価TEL下さい!!
Ⓑ Music studio PRO 68K (CZ-261MS) 定価￥28,800 特価TEL下さい!!
Ⓒ Print Shop V.2 (CZ-265HS) 特価TEL下さい!!
Ⓓ Multiword PRO 68K (CZ-225BS) 定価￥32,000 特価￥24,000
Ⓔ CZ-245LS (C-コンパイラII) 定価￥44,800 特価￥33,500
Ⓕ Telepotion PRO 68K (CZ-258BS) 定価￥22,800 特価￥18,000

## 通信販売のご案内

全国通販

■銀行振込で申し込みの方は商品名及びお客様の住所・氏名・電話番号をお知らせ下さい。
〔振込先〕第一勧業銀行 渋谷支店
普通No.1163457 ㈱オーエーランド
■現金書留で送金されるお客様は電話番号と商品名、数量を明記して同封して下さい。■クレジットでご購入を希望される方は申し込み用紙をお送り致しますのでご記入の上返送して下さい。20才以上の方は、原則として保証人不要です。クレジットは1〜60回払で月々5,000円よりご自由に設定できます。

東京門工学業校 東急観光 至三軒茶屋246号 大和證券 ヤマハエピキュラス 東急プラザ 京王井ノ頭線 歩道橋 南口バスターミナル JR渋谷南口 至恵比寿 ココです!! NEW OAランド (・ニコマートの2Fです。) 至六本木

■年中無休です!!

クレジット表

| 回数 | 率 | 回数 | 率 | 回数 | 率 | 回数 | 率 | 回数 | 率 | 回数 | 率 | 回数 | 率 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3回 | 3.5% | 6回 | 4.5% | 10回 | 6% | 12回 | 6% | 15回 | 8.5% | 18回 | 11% | 20回 | 12% |
| 24回 | 12.5% | 30回 | 17% | 36回 | 17.5% | 42回 | 22.5% | 48回 | 23% | 54回 | 29% | 60回 | 29.5% |

# ㈱オー・エー・ランド

〒150 東京都渋谷区桜丘町3-13 アルカディア2F
☎(03)3770-8855
関東エリアの送料は、1個につき￥1,000です。 FAX(03)3770-7080
★全商品保証書付。専門のアドバイザーが、お客様のニーズに対応します。
★初期不良・輸送トラブル等に迅速に対応し、即交換させていただきます。

■表示価格は、税別表示です。詳しくは、お電話にて、お問い合せ下さい。掲載の価格は、8月上旬現在です。

■店頭にて、新作ゲームソフト25〜30%OFF!!(税別)、超低金利オクトハッピークレジットをご利用下さい!!

■朗報デス。翌月末一括(10月末)払いOK!!手数料無料。ご利用下さい。■店頭にて、新作ゲームソフト25〜30%OFF!!

## パソコンプラザ

案内図

至品川　東口ロータリー　JR蒲田から徒歩5分　京浜蒲田から徒歩1分　至品川　三菱BK　JR蒲田駅　2F　商店街アーケード　京浜蒲田駅　富士BK　三和BK　メガネドラッグ　八幡神社　環八通り　NTT　〒　至横浜　至横浜

店頭セール実施中

## オクトで始まるパソコンワールド

☎03-3730-6271

●営業時間 AM 11:00〜9:00/日曜・祭日PM7:00　電話一本で、ハイ即納

〒144 東京都大田区蒲田4-6-7 FAX03-3730-6273

# 全国通販

●定休日毎週火曜日 祭日の場合翌日になります。

オクト ラクラククレジット

| 3回 | 3.5 | 6回 | 4.5 | 10回 | 6.0 | 12回 | 6.0 | 15回 | 9.0 | 18回 | 11.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 20回 | 12.0 | 24回 | 12.5 | 30回 | 17.0 | 36回 | 17.5 | 48回 | 23.0 | 60回 | 33.0 |

OCT-1 システム インフォメーション

- ▶全商品保証付(メーカー保証)
- ▶超低金利ハッピークレジット(1回〜60回)頭金ナシOK!
- ▶ボーナス一括払いOK!ボーナス2回払いOK!!
- ▶配達日の指定OK!(万全なサポート体制)
- ▶商品の組合せ自由! オクトフリーダムシステム
- ▶店頭デモンストレーション実施中

オクト セレクテッドシステム

広告掲載商品以外の製品も取扱っております。

■便利です　夜9時まで営業しております　お立寄り下さい　お待ちしております!!

蒲田

翌月末一括(10月末)払いOK!!手数料無料!!

# 過激に元気!年中夢中のオクトです!!

# XVI エクシヴィ

- ■16MHz■
- ■SX-WINDOW ver1.1■
- ■Attachment MEMORY BORD■

快速16MHz 鮮烈デビュー!!

### ■CZ-634C-TN (定価¥368,000)

Ⓐ●CZ-634C-TN
●CZ-614D-TN NEW
定価合計¥503,000 ▶特価TEL下さい。

| 12回 | ¥32,800 | 24回 | ¥17,400 | 36回 | ¥12,100 | 48回 | ¥ 9,500 |
|---|---|---|---|---|---|---|---|

Ⓑ●CZ-634C-TN
●CZ-607D-TN NEW
定価合計¥467,800 ▶特価TEL下さい。

| 12回 | ¥30,700 | 24回 | ¥16,300 | 36回 | ¥11,300 | 48回 | ¥ 8,900 |
|---|---|---|---|---|---|---|---|

Ⓒ●CZ-634C-TN
●CZ-606D-TN
定価合計¥447,800 ▶特価TEL下さい。

| 12回 | ¥29,200 | 24回 | ¥15,500 | 36回 | ¥10,800 | 48回 | ¥ 8,400 |
|---|---|---|---|---|---|---|---|

### ■CZ-644C-TN (定価¥518,000)

Ⓓ●CZ-644C-TN
●CZ-614D-TN NEW
定価合計¥653,000 ▶特価TEL下さい。

| 12回 | ¥42,600 | 24回 | ¥22,600 | 36回 | ¥15,700 | 48回 | ¥12,300 |
|---|---|---|---|---|---|---|---|

Ⓔ●CZ-644C-TN
●CZ-607D-TN NEW
定価合計¥617,800 ▶特価TEL下さい。

| 12回 | ¥40,400 | 24回 | ¥21,400 | 36回 | ¥14,900 | 48回 | ¥11,700 |
|---|---|---|---|---|---|---|---|

Ⓕ●CZ-644C-TN
●CZ-606D-TN
定価合計¥597,800 ▶特価TEL下さい。

| 12回 | ¥39,000 | 24回 | ¥20,700 | 36回 | ¥14,400 | 48回 | ¥11,300 |
|---|---|---|---|---|---|---|---|

## X68000XVI ドッカ〜ン!プレゼント!!

—あなたのオクトから素敵な贈物—

今、XVIをお買い上げいただいた方は、プレゼントの①番か②番のどちらかお選び下さい。プラス③番はもれなくプレゼント!!

▶現金超特価 ¥TEL下さい!!▶

① 生中継68 野球ゲームの決定版 (定価¥9,800)
大人気 大戦略II(キャンペーン版) 不朽の名作X68000版 (定価¥9,800)
新作!大人気ゲームソフト!!

or

② インテリジェントコントローラ
■CZ-8NJ2(CYBER STICK)
シューティングゲーマーの必須アイテム!!
(定価¥23,800)

③{MD-2HD(10枚) シリコンキーボードカバー もれなく!! サービス!!

※どちらかお選び下さい!!(どっちが得かヨーク考えてネ!)

## 特選周辺機器(送料¥500)

- ●SX-68M MIDインターフェースボード (システムサコム)¥19,800…特価¥13,800
- ●Fine Scanner X68(HAL研究所) (HGS-68)¥39,800………特価¥25,500

■増設RAMボード=I・Oデータ

① PIO-6BE1-A(1MB) ¥25,000…特価¥16,000
② PIO-6BE2-2M(2MB) ¥50,000…特価¥31,800
③ PIO-6BE4-4M(4MB) ¥88,000…特価¥55,000

## 周辺機器コーナー (送料¥500)

- ●CZ-6BEI IBM増設RAMボード……(¥35,000)▶特価¥26,000
- ●CZ-6BEIB IBM増設RAMボード……(¥28,000)▶特価¥21,000
- ●CZ-6BE2 2MB増設RAMボード……(¥79,800)▶特価¥60,000
- ●CZ-6BE4 4MB増設RAMボード……(¥138,000)▶特価¥103,000
- ●CZ-6BFI 増設用RS-232Cボード……(¥49,800)▶特価¥38,000
- ●CZ-6BGI GP-IBボード……(¥59,800)▶特価¥45,000
- ●CZ-6BMI MDIボード……(¥26,800)▶特価¥20,000
- ●CZ-6BNI スキャナ用パラレルボード……(¥29,800)▶特価¥22,500
- ●CZ-6BPI 数値演算プロセッサボード……(¥79,800)▶特価¥60,000
- ●CZ-6BOI ユニバーサルI/Oボード……(¥39,800)▶特価¥30,500
- ●CZ-6EBI/BK 拡張I/Oボックス……(¥88,000)▶特価¥65,800
- ●CZ-6VTI/BK カラーイメージ・ユニット……(¥69,800)▶特価¥52,000
- ●CZ-8NM2A マウス……(¥6,800)▶特価¥5,300
- ●CZ-8NTI マウストラックボール……(¥9,800)▶特価¥7,500
- ●CZ-8NSI カラーイメージスキャナ……(¥188,000)▶特価¥137,000
- ●CZ-6BCI FAXボード……(¥79,800)▶特価¥60,500
- ●CZ-8TM2 モデムユニット……(¥49,800)▶特価¥38,000
- ●CZ-64H 増設ハードディスク……(¥120,000)▶大特価
- ●CZ-6TU GY/BK RGBシステムチューナー……(¥33,100)▶特価¥24,500
- ●BF-68PRO 高性能CRTフィルター……(¥19,800)▶特価¥15,000
- ●CZ-6MOI 光磁気ディスクユニット……(¥450,000)▶特価¥328,000
- ●CZ-6BSI SCSIインターフェースボード……(¥29,800)▶特価¥22,200
- ●CZ-6BL2 LANボード……(¥298,800)▶特価¥220,000
- ●CZ-6BVI (ビデオボード)……(¥21,000)▶特価¥15,500
- ●CZ-6BE2A 2MB増設RAMボード……(¥59,800)▶特価¥43,800
- ●CZ-6BE2B 2MB増設メモリ(チップ型)……(¥54,800)▶特価¥40,000
- ●CZ-6BP2 数値演算プロセッサ……(¥45,800)▶特価¥34,000
- ●AN-S100 スピーカーシステム(2本1組)……(¥36,600)▶特価¥27,000

※クレジットの回数は1回〜60回、ボーナス併用などありますのでお電話でお問合せ下さい。

■本体セット:送料無料 (注)本体セット以外の周辺機器(プリンター、モデム、HDD等)及びソフトの送料は、北海道・九州地区=1ケロ¥1500、■その他離島地区は、1ケロ¥2000となります。

※上記料金には、消費税は含まれておりません。消費税が付加されますので、詳しくは、電話でお問合せ下さい。

~84回払いまでOK.!!

★頭金なし!★即日発送

●価格は流通事情により変動致しますので、銀行振込・書留等の送付前に、あらかじめお電話にてご確認下さい。

# P&Aがズバリ超特価セールでご奉仕!!

立寄り下さい。専門係員が説明いたします。
価で受付します。詳しくは電話にてお問合せ下さい。
価の20%引きOK! TELください。

# 全国通販

## X68000用ソフトコーナー (送料1ヶ~5ヶまで¥500・消費税別)

●Z's STAFF PRO68K Ver.2.0(ツァイト)……定価¥ 58,000➡特価¥ 38,000
●Z's TRIPHONY デジタルクラフト(ツァイト)……定価¥ 39,800➡特価¥ 27,800
●テラッツォ(ハミングバード)……定価¥ 19,400➡特価¥ 14,200
●KAMIKAZE(サムシング・グッド)……定価¥ 68,000➡特価¥ 44,800
●C & Professional Pack(マイクロウェアジャパン)……定価¥ 58,000➡特価¥ 41,000
●Final Ver3.2(エーエスピー)……定価¥ 38,000➡特価¥ 29,600
●C-compiler PRO68K Ver.2 CZ-245L……定価¥ 44,800➡特価¥ 33,300
●CARD PRO68K CZ226BS……定価¥ 29,800➡特価¥ 21,200
●XBAS to C CHECKER CZ-260LS……定価¥ 9,800➡特価¥ 7,400
●OS-9/X68000 CZ219SS……定価¥ 29,800➡特価¥ 22,500
●AI-68K CZ234LS……定価¥188,000➡特価¥138,000
●THE 福袋 V2.0 CZ224LS……定価¥ 9,900➡特価¥ 7,400
●SOUND PRO68K CZ-214MS……定価¥ 15,800➡特価¥ 11,400
●MUSIC PRO68K CZ213MS……定価¥ 18,800➡特価¥ 13,400
●Sampling PRO68K CD215MS……定価¥ 17,800➡特価¥ 12,700
●MUSIC-studio PRO68K CZ-252MS……定価¥ 15,800➡特価¥ 21,400
●MUSIC-PRO68K〔MIDI〕247MS……定価¥ 28,800➡特価¥ 20,700
●New-print Shop 221HS……定価¥ 19,800➡特価¥ 15,500
●Communication 223CS……定価¥ 19,800➡特価¥ 14,200
●Communication Ver.2 CZ-257CS……定価¥ 19,800➡特価¥ 15,500
●C-TRACE68 Ver.3.0(キャスト)……定価¥ 98,000➡特価¥ 69,000
●サイクロンEXPRESSα68……定価¥ 98,000➡特価¥ 69,800
●G68K Ver2 PRO……定価¥ 22,000➡特価¥ 17,500
●SX-WINDOW CZ-259SS……定価¥ 6,800➡特価¥ 4,900
●Gツール(ザインソフト)……定価¥ 28,000➡特価¥ 18,900
●たーみのる2(SPS)……定価¥ 17,800➡特価¥ 13,300
●マジックパレット(ミュージカルプラン)……定価¥ 19,800➡特価¥ 14,500
●Hyper word CZ-251BS……定価¥ 39,800➡特価¥ 29,600
●ゲームソフト20%OFF OK!!(一部ソフト除く)

## 周辺機器コーナー (送料¥500・消費税別)

1 CZ-8NSI……定価¥188,000▶特価¥140,000
2 CZ-6VTI……定価¥ 69,800▶特価¥ 52,500
3 CZ-6TU……定価¥ 33,100▶特価¥ 24,500
4 BF-68PRO……定価¥ 19,800▶特価¥ 15,300
5 CZ-6BEI……定価¥ 35,000▶特価¥ 26,000
6 CZ-6BEIA……定価¥ 38,000▶特価¥ 28,600
⑦ CZ-6BE2A……定価¥ 59,800▶特価¥ 44,200
⑧ CZ-6BE2B……定価¥ 54,800▶特価¥ 40,800
9 CZ-6BFI……定価¥ 49,800▶特価¥ 38,200
10 CZ-6BPI……定価¥ 79,800▶特価¥ 60,000
11 CZ-6BMI……定価¥ 26,800▶特価¥ 20,300
12 CZ-6EBI……定価¥ 88,000▶特価¥ 66,500
13 AN-S100……定価¥ 36,600▶特価¥ 28,500
14 CZ-6SDI……定価¥ 44,800▶特価¥ 35,000
15 CZ-6BN1……定価¥ 29,800▶特価¥ 22,600
16 CZ-6BV1……定価¥ 21,000▶特価¥ 15,900
17 CZ-64H……定価¥120,000▶特価¥ 91,500
18 CZ-6BG1……定価¥ 59,800▶特価¥ 45,000
19 CZ-6BU1……定価¥ 39,800▶特価¥ 30,300
20 CZ-6PVI……定価¥198,000▶特価¥153,000
21 CZ-6BS1……定価¥ 29,800▶特価¥ 22,300
22 CZ-8NJ2……定価¥ 23,800▶特価¥ 18,500
23 CZ-6BL2……定価¥298,000▶特価¥222,000
24 JX-100S……定価¥ 89,800▶特価¥ 48,500
㉕ JX-220X……定価¥168,000▶特価¥126,000
㉖ IO-735XB……定価¥248,000▶特価¥169,000
(IO-735XBご購入の方「BANANA-PRINT」プレゼント!!

## X68000用ハードディスク (送料¥1,000)

アイテック

■TX-80(80MB)……定価¥108,000▶特価¥ 78,000
(SCSI・SASI両用) (送料・消費税込み¥81,370)

■TX-130(130MB)……定価¥138,000▶特価¥ 98,000
(SCSI) (送料・消費税込み¥101,970)

■TX-180(180MB)……定価¥185,000▶特価¥132,000
(SCSI) (送料・消費税込み¥136,990)

## プリンター(ケーブル・用紙付) (送料¥1,000・消費税別)

■CZ-8PC5-BK NEW……定価¥ 96,800▶特価価格はTEL.!!
■CZ-8PK10……定価¥ 97,800▶特価¥71,000
■CZ-8PG2……定価¥160,000▶特価価格はTEL.!!
■CZ-8PG1……定価¥130,000▶特価価格はTEL.!!

## モデムコーナー (送料¥1,000)

■COMSTARZ CLUB24/5
(NEC)定価¥39,800
特価¥26,500 (送料・消費税込み ¥28,325)

■MD-24FB5V
(オムロン)定価¥39,800
特価¥26,500 (送料・消費税込み ¥28,325)

## P&A特選パソコンラック (消費税別)(送料無料)

①3段¥8,800 ②4段¥9,800 ③5段¥13,800

全機種=移動自由(キャスター付)・キーボード収納可(5段のみ)=1230(H)×600(D)×650(W)

超特価でクレジットが組める!!

## 中古パソコンはP&Aにおまかせ!!

その場で高価現金買取り・高価下取りOK!!

■まずはお電話下さい。
03-3651-1884、FAX:03-3651-0141

■下取り・買取りでお急ぎの方、直接当社に来店、または、宅急便にてお送り下さい。

●下取りの場合……価格は常に変動していますので査定額をお電話で確認して下さい。(差額は、P&A超低金利クレジットをご利用下さい。)

●買取りの場合……現品が着き次第、2日以内に買取り金額を連絡し、振込み、又は書留でお送り致します。

●近郊の方は、P&A本店まで、直接お持ち下さい。即金にて、¥1,000,000までお支払い致します。

## 《便利な超低金利クレジットをご利用下さい》

●月々¥1,000円からOK!! ●ボーナス払いOK(夏冬10回までOK)
●支払い回数 1回~84回 ●お支払いは、8ヶ月先からでもOK!!

## 通信販売お申し込みのご案内

〔現金一括でお申し込みの方〕
●商品名およびお客様の住所・氏名・電話番号をご記入の上、代金を当社まで、現金書留でお送りください。(プリンター・フロッピーの場合、本体使用機種名を明記のこと)

〔銀行振込でお申し込みの方〕
●銀行振込ご希望の方は必ずお振込みの前にお電話にてお客様のご住所・お名前・商品名等をお知らせください。(電信扱いでお振込み下さい。)

〔振込先〕 住友銀行 新小岩支店
普通預金 1451576 ㈱ピー・アンド・エー

〔クレジットでお申し込みの方〕
●電話にてお申し込みください。クレジット申し込み用紙をお送りいたしますので、ご記入の上、当社までお送りください。
●現金特別価格でクレジットが利用できます。残金のみに金利がかかります。
●1回~84回払いまで出来ます。但し、1回のお支払い額は¥1000円以上。

### 超低金利クレジット率

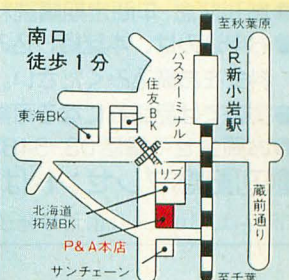
| 回数 | 3 | 6 | 10 | 12 | 18 | 24 | 36 | 48 | 60 | 72 | 84 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 手数料 | 3.5 | 4.5 | 6.0 | 6.0 | 11.0 | 12.5 | 17.5 | 23.0 | 29.5 | 38.0 | 45.5 |

アフターサービス万全
全商品保証付。専門の担当者がお客様の立場で対応します。
初期不良、輸送トラブルetc.
万が一初期不良、輸送トラブルが発生しました際には、即交換させていただきます。

●定休日/毎週水曜日=第3水曜(祭日の場合は翌日になります)

マイコン 専門 ショップ

株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目1番地19号
☎03-3651-0148(代) FAX. 03-3651-0141

営業時間
平日:AM10:00~PM7:00
日祭:AM10:00~PM6:00

●現金書留及び銀行振込でお申し込みの方は、上記商品の料金に3%加算の上でお申し込み下さい。詳しくは、お電話でお問い合せ下さい。

SOFT
BANK

［第5回］
いろいろな形

寺尾 響子

# 響子inCGわ〜るど

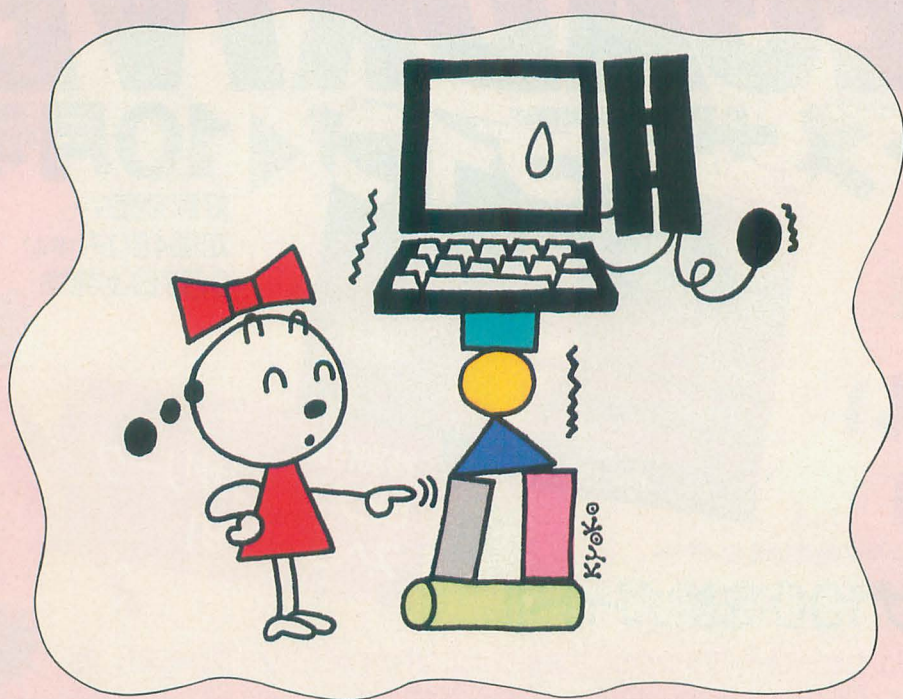
遠い記憶になろうとしている湾岸戦争のニュースで，いまでもはっきりと覚えている映像があります。ステルス爆撃機ノースロップB-2。レーダーに映らない特殊な塗料を施した黒いボディは，ブーメランをふくらませたような無駄のない美しいラインでできていました。「なんてきれいな形なんだろう」と思った次の瞬間，戦争の道具に美しさを感じたことが嫌になりました。

## CGのなかの形

CGで表現してみたい形はたくさんあります。ロボットやバイク，なめらかな人体，雲，トータルリコールに出てくるような近未来都市などなど。こうした形をつくるために，3次元のCGではさまざまな工夫がされてきました。X68000ではどんなことができるでしょうか。

方眼紙でペーパークラフトをつくるのによく似たCGがあります[1]。まず，3つ以上の点をぐるりとむすんで多角形をつくります。一枚の面ができました。この面を何枚も立体的に貼り合わせるようにして形をつくってゆきます。多角形でつくった面のことをポリゴンといいます[2]。コマーシャルやニュースのオープニングなどのCGは，ほとんどがポリゴンで形づくられています。

積木のような形を使ってつくるCGもあります[3]。今回のCGでUFOの間に散らばっている色とりどりの物体がその基本形で，ソリッドモデルといいます。これらの物体をくっつけたり，ある物体を別の物体で切り取ったりしてデザインをします。感覚としては，自分でプラモデルのパーツを削ってつくり，組み立てていくのに近いでしょう。

流れるような，なめらかな曲面をつくるのに向いているのがメタボール[4]。とても魅力的な形です。まえに，水銀体温計を過って落としてしまい，水銀が床にこぼれたことがありました。水銀の玉どうしは近づけると，くるんとひとつにまとまります。少しはなすと，ひょうたんのようにつながります。メタボールの使いごこちは，この水銀で形をつくっているような感じがしました。

メタボールを使わなくても，ある程度のなめらかな表現はできます。ポリゴンでつくった形の凹凸をとるスムースシェーディングという処理があるのです[5]。また，ソリッドモデルはそれ自体が曲面をもっているので，つるりとした表現にも向いているでしょう。

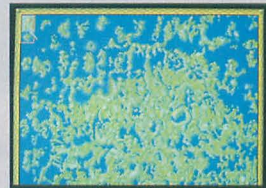
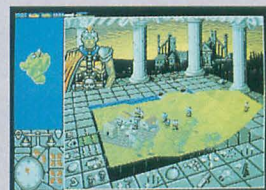
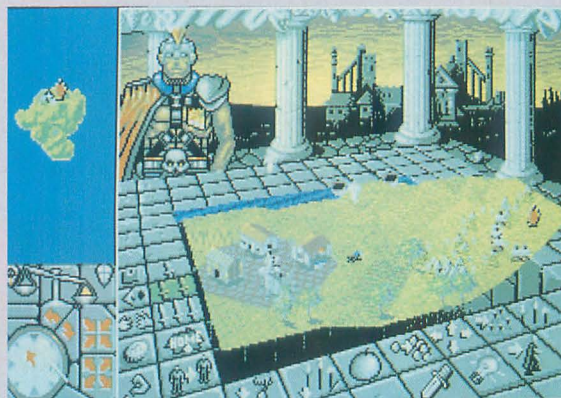
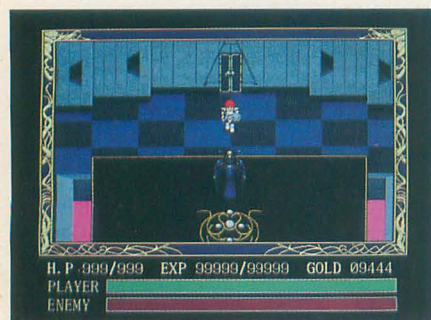
今，手に入るソフトウェアでこんなにいろいろな形をつくることができます。まわりを見渡してちょっと面白いデザインを見つけると，CGでどうやってつくろうかなと考えてみます。パズルを解くようでなかなか楽しいものです。

物をつくるというのは，いままでにあった形をいったんこわして新しく組み直す作業だと思っています。すでにあるものでも分解し再構築することで，まったく別のものに生まれ変わります。物体のデザインだけでなく，文章を書く，作曲をする，ゲームデザインをするなど，すべてのつくることに共通していえるのではないでしょうか。

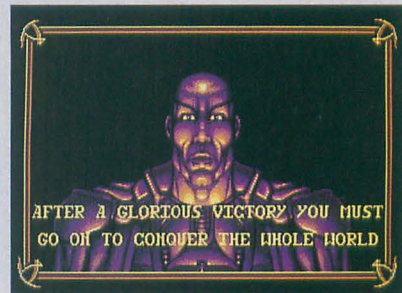
---

1) この方法でつくるものに，MAGIC，DōGA，Z'sTRIPHONYがあります。
2) ポリゴンの面を塗りつぶさずに線だけで立体を表したのがワイヤーフレームです。
3) C-TRACE，サイクロンがそうです。
基本形の半径，高さ，一辺の長さなどは自由に変えられます。また，カクカクした球体はポリゴンでつくったもの。
4) C-TRACE＋で扱えます。連載第2回の犬のキャラクターはメタボール22個でできています。トランスピュータがあっても犬1匹つくるのに1日中かかってしまいました。
5) DōGAのCGAシステムやZ'sTRIPHONYで可能です。
6) ポリゴン，ソリッドモデル，メタボールのほかにフラクタルモデルというのがあります。山や海岸線などの自然の景観をつくるのに使われます。

# SOFTWARE INFORMATION

3D2が正式名称「スターウォーズ」となり，ビクター音楽産業から発売になったというのが，今月いちばんの大ニュース。新作紹介の最後のほうで紹介しているのでお見逃しなく。あとは海外ソフトからの移植作，「パワーモンガー」と「ドラッケン」が注目でしょうか。

## パワーモンガー

「ポピュラス」を作ったピーター・モリニュー氏の新作ということで，話題になった「パワーモンガー」が移植された。イマジニアからはほかにも「シムアース」などが移植予定にラインアップされているが，X68000ではこの「パワーモンガー」がいちばん早い発売となる。肝心の内容のほうは「ポピュラス」とは少し違っていて，ストラテジーに富んでいる。

最初のほうの面では，羊と村を襲撃していれば，すなわち，戦いのコマンドだけを使っていれば，なんとなく進行していくのだが，面が進むとそうもいかない。武器の発明，同盟，敵の武将を手下にするなどという，どこかのゲームのようなことも必要になってくるのである。とはいえ，すべてリアルタイムで進行するゲームなのはたしか。今回届いたサンプルでは音などが入っていなかったが，ほかのところはそこそこできていたので，発売は近いのでは？

X68000用　5″2HD版　価格未定
イマジニア　☎03(3343)8911

### イースの伝説は生きていた

| 順位 | タイトル | (前回順位) |
|---|---|---|
| 1. | イース | 3 ↑ |
| 2. | パロディウスだ！ | 2 |
| 3. | ファランクス | 1 ↓ |
| 4. | 生中継68 | 4 |
| 5. | 遥かなるオーガスタ | 6 ↑ |
| 6. | スターウォーズ | ―初 |
| 7. | 信長の野望・武将風雲録 | ―初 |
| 8. | グループ・エックス | ―初 |
| 9. | ボナンザブラザース | 10 ↑ |
| 10. | ロードス島戦記 | ― |

やあ。元気だったかナ。ジーザスIIを見ていたら五色和也の話し方がうつってしまった浦川です。アハッ。

今月トップを奪ったのは，「ファランクス」でも「パロディウスだ！」でもなく，「イース」でしたネ。グラフィックのタッチは意見が分かれるかと思ったけど，意外に評判はよかったみたい。みんな好き嫌いはあっても，X68000のグラフィックパワーを存分に活かしているのが魅力的に映ったみたいだナ。

「パロディウスだ！」もしっかり踏みとどまりましたネ。“「ファランクス」は難しすぎる！”とか“「ファランクス」より面白い！”とか，いやに意識したハガキが多いゾ。まあ同じシューティングだし，わからないこともないケド。「ファランクス」のほうも「パロもいいけど，こっちのほうがスカッとする」とか「シューティングの頂点だ！」とか，熱の入ったハガキがたくさんあるナ。

チャートの中にはスポーツシミュレーションが多いネ。4位キープの「生中継68」，それを追いかけるのが「遥かなるオーガスタ」，8位初登場の「グループ・エックス」もモータースポーツといっていいのかナ。「グループ・エックス」には「ザ・コックピットを思い出す」，「自分の車が入ってるかどうか気になる」なんて声があるネ。

今月の注目株はなんといっても「スターウォーズ」だ。正式名称が決まる前のハガキだから，「3D 2」になってるけど，いきなりチャートに入っちゃいました。M.N.M.は「マジカルショット」に続いてのランクインだネ。期待どおりのデキで，時期がよければひょっとしてトップなんてことも十分ありうるナ。注目してみよう。

もうひとつ初登場は信長の野望シリーズ最新作，「武将風雲録」。「なんだかんだいっても遊んでしまう作品」「長く遊べるという点ではピカイチだと思う」という声が。まあ，定番ですからランクインも当然ですネ。

さて，6位までどれもトップに行けるパワーのあるソフトが押し並ぶなか，来月の1位を制するのはどれか！　また来月ぅー。　(浦)

## ドラッケン

突如，神の大いなる意志により破滅の道を辿ることになった人類。次にこの世を支配するのはドラゴン型人類「ドラッケン」なのか？　人類は世界より選にりすぐられた4人の戦士にすべてを委ねる。

フランス生まれのリアルタイムRPGゲーム「ドラッケン」がついに登場。3D処理による全方向への高速移動，刻々と変化する地形と空，迫りくるデカキャラ。かつての日本製のゲームには見られなかった斬新なアイディアと奇抜な視点で綴られた長編大作。

PC-9801版と比べて，グラフィックやサウンドなどがさらに強化。24kHzモードが用意されているのもいい。ダンジョン内での操作性が少々複雑なのが難といえば難だが，これには慣れるしかない。

とにかく，ひさびさに長く遊べそうなRPGのような気がする。　(善)

X68000用　5″2HD版2枚組　9,700円(税別)
エピック・ソニー　☎03(3475)2632

## 機動戦士ガンダム クラシック・オペレーション

機動戦士ガンダムといえば，テレビ放映を重ねていくうちにどんどん人気を集め，最終的には国民的な人気(というと，少しいいすぎかな)となってしまった名作アニメ。このガンダムをゲーム化するというのは，パソコンゲーム創世以来，さまざまなジャンルで何度となく行われてきた。

そのうちのひとつ，昨年，他機種で発売された「クラシック・オペレーション」がX68000に大幅にパワーアップして移植される。ビジュアルシーンの全面的な書き直し，PC-9801の「デザート・オペレーション」のシステムの組み込みなどが主な特徴だ。

また，モビルスーツに搭乗できるキャラクターは24人以上から選ぶことができる。集中攻撃システム，キャラクターネーミングシステム，ストレスシステム搭載。軍備の編成は2つの年代のものから選択できる。

X68000用　5″2HD版　9,800円(税別)
ファミリーソフト　☎03(3924)5435

## ゼノン2

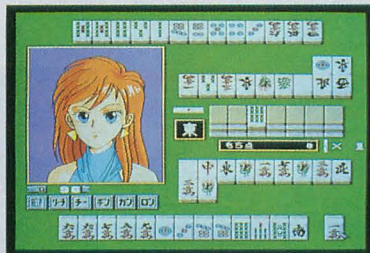
一部のAMIGA通に親しまれているといわれる，ビットマップブラザーズの「XENON2」がいよいよ移植の運びとなった。ビットマップと社名に入っているだけあって，この会社のグラフィックには相当の自信がうかがえるが，移植版のこの「XENON2」でもそのグラフィックはなかなかのものであるといえよう(特に質感)。ここまできれいならば音楽にも期待が高まるが，オリジナルよりも曲数を増やしてバリバリになるとのこと。いまの段階では残念ながらまだ聞くことはできなかったが，期待に応えるだけのモノであることは間違いなさそうだ。肝心のゲーム内容は，オーソドックスな縦スクロールシューティングゲームであり，敵を倒してショップで武器を買い，面の最後はボスと対決といった構成で誰にも親しみやすいものになっている。制作者のこだわりとセンスが伝わってくるこの1本，日本のゲームに飽きてしまった通な人に一度試してもらいたいソフトである。　(八)

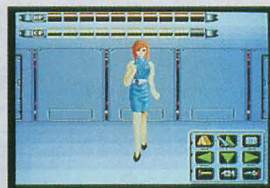
X68000用 5″2HD版　価格未定
エピック・ソニー　☎03(3475)2632

## 麻雀マスター

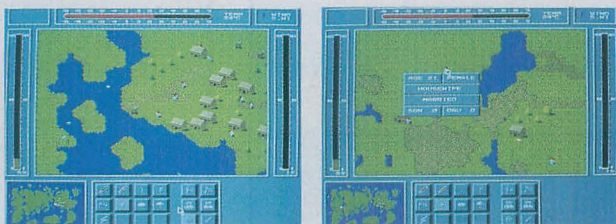
この3Dダンジョン麻雀ゲーム「麻雀マスター」は大阪のソフトハウス「アレックス」が開発，TAKERUオリジナルとして発売される。

洋上に浮かぶ巨大なプラントタワー。その中では，哨戒監視用のロボットや，ガーディアン兼メイドとして働く女性アンドロイドの開発が行われていた。ところが，ここを集中管理するコンピュータが逆らうようになり(よくある話)，人間たちは追い出されてしまった。

で，ここの破壊工作を頼まれたのがプレイヤーということになるのですが，アンドロイドはなかなか手強い。しかし，アンドロイドの弱点，すなわち麻雀に負けると，テレポートして逃げたり，無抵抗になり★★★しまう(なんじゃそりゃ，まあ，だいたいわかるけど)ということがわかったのであった……。

とりあえず，ダンジョン型ロールプレイングの戦闘部分が麻雀になったもので，「脱ぐ要素も多少あり」とのこと。ボイス演出やアニメーションもばっちりだそうだ。

X68000用　5″2HD版2枚組　9,800円(税別)
ブラザー工業(TAKERU)　☎052(824)2493

## ノア

「スターウォーズ」の開発で熱い視線が注がれているM.N.M.Softwareから，ちょっと変わったゲームが発売される。環境保全シミュレーションゲームとでもいうべきゲームで，「ノア」という名前だ。

プレイヤーは気温，天候，風向きなどを変化させることで，世界のバランスをとりながら，人間が環境を破壊するのを食い止める。“それじゃあ，人間を全滅させればいいのでは？”という疑問もわいてくるかもしれないが，そうもいかない。人間を生かしつつ，この世界の調和を保とうというのが目的なのである。なにもしなくても世界は変化していくので，眺めているだけということもあるが，それも狙いのひとつらしい。

X68000用　5"2HD版　7,200円(税別)
M.N.M.Software　☎0423(60)3084

## コラムス

このゲームはいわゆるブロックを落としていくタイプのパズルゲール。ブロックは縦3つに連なっていて，3つそれぞれに色の異なる宝石が入っていて，同じ色の宝石が縦，横，斜めのいずれかに3つ以上並べば消える。宝石が消えたところには，上に乗っかっていた宝石がずれてくる。

ブロックがただ単に落ちてくるだけではなかなか消えないので，ブロックを左右に動かしたり，中の宝石の順番を入れ替えたりしてやって，よきにはからってやる。ブロックの山が上まで到達してしまってどうしようもなくなるまで延々と続く，麻薬性があるゲームだ。面を次々とクリアしていく「ステージモード」や「対戦モード」なども用意されている。

X68000用　5"2HD版　7,800円(税別)
システムソフト　☎092(752)5278

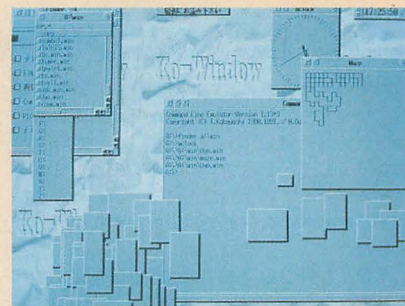
## TAKERU関連諸々

「Ko-Window」アプリケーション集Ｉが発売される。ネットでも公開されているものだが，ソースリストも付属。Ko-Window ver.2.24+5も入っている。内容は，

| | |
|---|---|
| ファイルセレクタ | finder.Win |
| スクリーンエディタ | KE.win,stevie.win |
| ゲーム | TaTRIS.win |
| 多機能時計 | timer.win |
| 音楽演奏 | PLAY.win |
| | KoMXP.win,KoRCP.win |
| デモ | Uyo.win,bug.win |
| | MAZE.win |

このほか，TV.win,GPIC.win,CutEdit.win,ebox.win,COM.win,CODE.winなど。Ko-Windowの操作方法を説明してくれるプログラムや，アプリケーション開発キットも付属している。価格は1,600円。

また，ターボコンソール用明朝体漢字フォントが5,800円，電脳音楽・クラッシックの巻（その1）が2,000円（すべて税込）で発売される。

ブラザー工業(TAKERU)　☎052(824)2493

## 「ストリートファイターII」大会観戦記

カプコンの「ストリートファイターII（通称ストII)」といえば，知らない人でも知ってる大ヒットアーケードゲーム。わたくし浦川も毎日毎日プレイするのに30分も並び続け，サボッた授業は数知れずというていたらくです。

8月6日にゲーム雑誌「ゲーメスト」の主催でストIIの対戦大会が開催されると聞いて，全国のレベルを窺い知るべく，いそいそと出かけていくのは当然のことともいえるでしょう。

案の定，出場者の募集はすごい人気だったそうで，500名の定員に2000人も押しかけたという話。結局，抽選で選んだということです。でも500人といってもあーた，100人が5つですよ，相当大がかりなイベントには変わりない。

サンシャインシティ文化会館4Ｆに設けられた会場には16台の筐体が設置され，そのうち4台の映像は100インチプロジェクタでも観戦できるようになっていました。1回戦から結構いい試合が多くて，逆転また逆転の連続。ザンギエフのようなマイナーなキャラクターが登場するとみんな押しかけて一緒に応援していました。いやあ，100インチで見るスピニングパイルドライバーはすごかったな。

500名の選手はトーナメントで徐々に絞り込まれ，勝ち残っている証の赤ハチマキもみるみるうちに減っていきます。

もともと見てても楽しいゲームのうえに，ギャラリーがみんなエキスパートだから，本当にいいプレイには惜しみない拍手や歓声が巻き起こっていました。

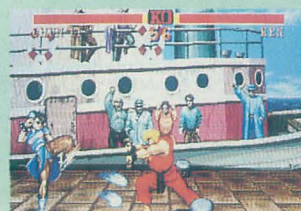
午後には特別プログラムとして，ゲーメスト編集部VSカプコン，選手代表VSカプコンの対戦も行われました。カプコン開発陣の「しょ，しょ，しょーりゅーけん」コールが楽しかった。やたら昇竜拳にこだわって，見せるプレイに徹していたようです。そのせいかあまり成績のほうはよくなかったみたいだけど。

そしていよいよベスト8の対戦。4面のプロジェクタに映し出された映像をみんな食い入るように見つめています。さすがに戦いのレベルは高く，ジャンプした敵は正確にたたき落とすわ必殺技はかわすわで1秒先がどうなるのか，予測のつかない戦いが繰り広げられました。

なんと決勝戦は2人ともガイル！　ガイルを使えないときのプレイのうまさが明暗を分け，東京の清藤幸治君が見事優勝，マウンテンバイクを手にしたのでした。

私も勝ち抜き戦の筐体にちょっと挑戦してみましたが，得意キャラが選べなくてアッサリ負けてしまいました。うーん，春麗なら勝てたのに（負け惜しみ）。

くそー，こんなに面白いイベントだったら，第2回があったときには私も参加しようかな。

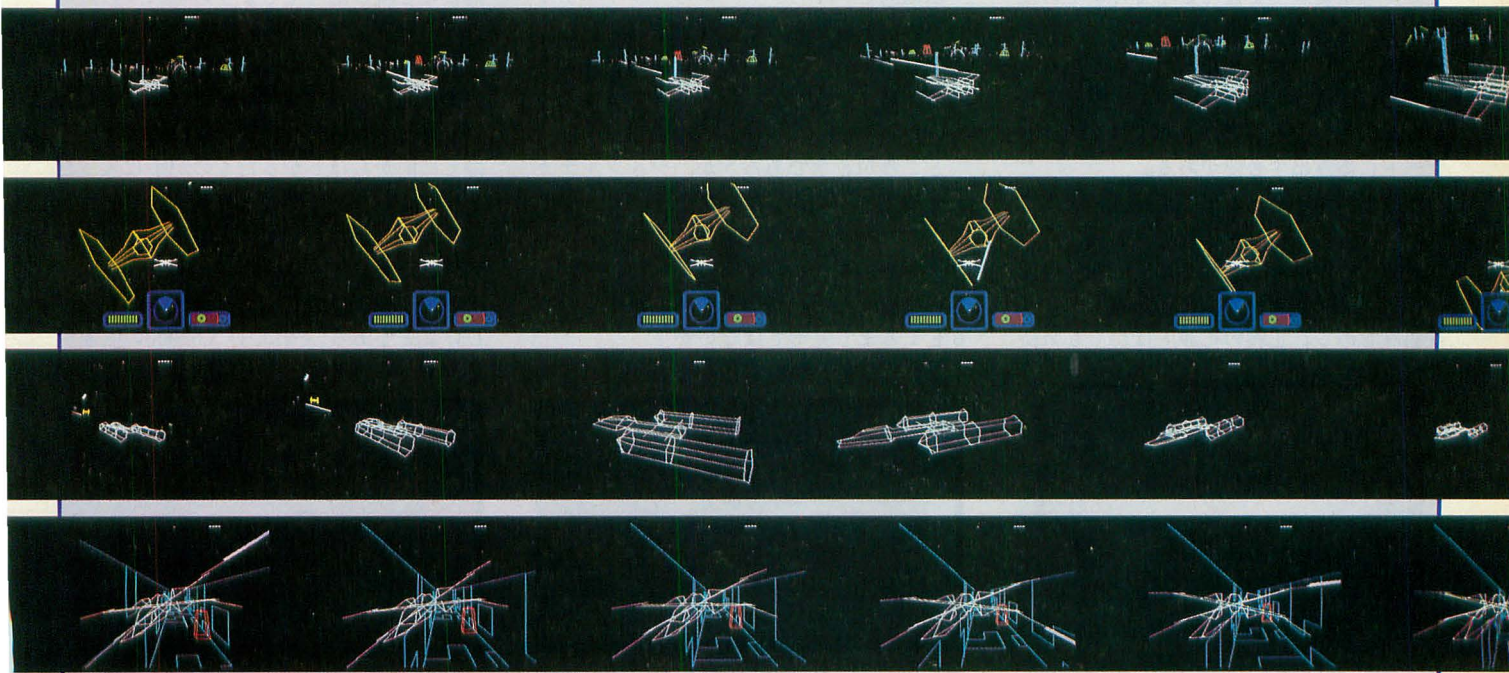
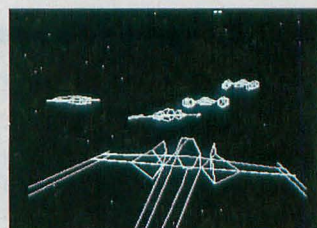
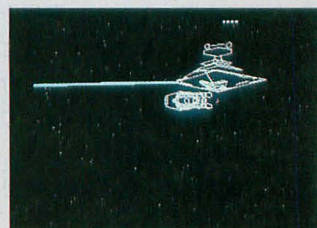
（浦）

# スターウォーズ

3D2(仮称)。まるでヒーローもののテレビ番組の3クール目になって突如現れる正体不明のキャラクターのように謎に満ちていたゲームの正体がついに明らかになった。なんと，あの「スターウォーズ」のゲーム化だったのだ！ デモを見よう。映画そっくりの出だしからデススター攻略のミッション解説がアニメーションで示される。ゲーム展開を見る。Xファイターを駆ってタイファイターを蹴散らし，要塞に突入。地表面では砲台を破壊しつつ，表面溝への入り口を探す。そして，溝の中では追手を振り切り，デススター中枢に爆弾を投下する……という具合。スターウォーズ第1作の山場を忠実に再現している。

見てのとおり，内容はシューティングゲームだが，3Dであること以外にも処理自体はシミュレーションの様相が強い。まず，普通のシューティングゲームと違ってプレイヤーひとりで戦うのではない。味方機も独自に敵機と空中戦を繰り広げる。ピンチに助けられることもある。といってもこれらは演出されたものではなく，各機のアルゴリズムがもたらす筋書きのないドラマのひとつにすぎない。

ゲーム内容とは関係ないが，なぜか安い！ 以前，パナソニックがジョージ・ルーカスとスターウォーズのキャラクターをCMで使ってかなりふんだくられたという話は有名だ。また，長らくタイトルを公表できなかったり，急遽ビクター音楽産業からの発売になるなど権利関係の複雑さを暗示するものも多い。これでどうして，安いのか？ 以前DIME誌で，ルーカスフィルムは映画だけでなく総合エンタテイメントコングロマリットを目指すという記事があった。自らコンピュータソフトに参入したのもその関係が強く，巨大な目標に向かっていくため，いくつかの企業(THXシステムで松下など)と提携して事業を進めているという。今回の驚異の低価格が実現されたのも，M.N.M.ソフトウェアがルーカスグループのビジネスパートナーとして認められたからといってもいいだろう(無論M.N.M.のポリシーも関係するが)。

今回のスターウォーズは単にライセンスのみでなく，ルーカスフィルムの全面的な協力のもとに作成されている。音声は専用トラックから取られるので，台詞の後ろに映画のBGMが流れていたりということはない。BGMにはもちろんスターウォーズの曲が使われる。あの宇宙に響くシンフォニックサウンドが内蔵音源で再現できるのか？ という懸念ももっともだが，音楽は「古代祐三の本気モード」だ（アクトレイザーもまだ彼の全力ではないという)。期待しよう。 (S.N.)

X68000用5"2HD版 7,200円(税別) ビクター音楽産業 ☎03(3423)7901

# 嘘つきは泥棒のはじまり

Yaegaki Nachi
八重垣 那智

「♪俺たーちゃ，ボナンザブラザーズ」という歌声にのって２人の泥棒が現れた!?　麻酔銃を手に，つぎつぎと悪の建物に侵入していく。トボケたこともするけれど，正義の泥棒はやっぱりみんなの人気者だよね。

泥棒は悪いことである。たしかに，人様のものを勝手に自分のものにしてしまうのだから，悪いことには違いがない。しかし泥棒はかっこいい職業である。

もちろん名刺の片隅に「職業　泥棒」と書いて配ることはできないが，古くはアルセーヌ・ルパンの時代から痛快無比な冒険を繰り広げ，高笑いとともに警察を煙に捲くのは泥棒の特権として伝えられてきた。

しかし，誰でも泥棒になれるわけではない。人々は皆正直に生きているので，しかたなく小説や映画や漫画，アニメといったものにその身をなぞらえて，ため息をつくのである。

## 泥棒のススメ◆◆◆◆◆◆◆◆◆◆◆◆◆

というわけで，この「ボナンザブラザーズ」は，人々の永遠の夢を実現した，いわば「泥棒シミュレータ」である。

悪の町バッドタウンの人々を救うため，正義の泥棒として各地の悪の温床に潜入し，そこから重要証拠を盗んでくるのがその使命だ。

しかし，自分たちが正義の泥棒であることを忘れてはいけない。いかに厳重な警備がなされていようと，ひとりも殺さずにスマートに盗み出す必要がある。

すべての品物を盗み出して屋上に脱出すれば，飛行船に飛び乗って悠々と逃げられるといった段取りができており，悪者の間にボナンザブラザーズの名前が轟くという仕組みである。正義の犯罪とはかくも気持ちのイイものなのだ。

２人同時プレイにもいうまでもなく対応している。彼ら２人は別行動でもなんでも自由自在の泥棒コンビである。そのため上下をそれぞれのプレイヤー用の画面に振り分け，それぞれが独立して盗みのできるゲーム構成を取っている。真ん中には建物のマップがあり，より戦略性が高まるというわけである。

ボタンは敵を気絶させる麻酔銃とジャンプという組み合わせになっているので，そんなに違和感はない。近頃の泥棒は合理的なのがモットーなのである。

実はこのゲームのオリジナルは，セガから1990年（去年だな）に発売されたアーケードゲームである。アーケードにはめずらしくハイレゾグラフィックを搭載したシステム24というハードによって，このあまりにも綺麗でかつ笑える独特の世界を作り上げていた。目で見て印象に残るゲームだったといえるだろう。

私は同じハイレゾゲームの「サイバリオン」の解像度があっさり変更されていたことを忘れていなかったので，このゲームも同じ運命に遭うものと決めつけてあきらめていた。

ところがどうであろうか，ツルツルテカテカのキャラクターたちはたまにカッコよく，ふだんはちょっと頼りなく，ゲームセンターそのままに動き回っていたのである。さらには，止まっているとハエが飛んでくるところまで完璧ではないか！

これには気が動転してしまい，思わず開発元のSPSがある福島の方角に足を向けて，３日３晩寝込みそうになってしまったほどであった。とにかく似ている。しかも似ているのはそれだけではなかったのである。

## 泥棒はつらいよ◆◆◆◆◆◆◆◆◆◆◆

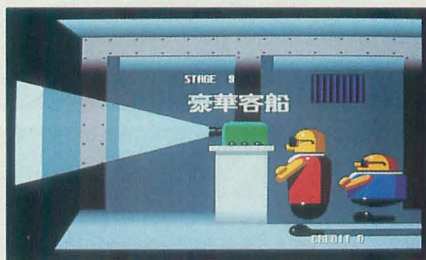
当時，ゲームセンターで泥棒の修業をするため，せっせと「ボナンザブラザーズ」にはまっていた私であったが，とてもとても大泥棒にはほど遠く，逮捕されて苦汁を舐める日々が続いていた。たしかに実際の泥棒は難しいし，ゲームだからといってそれが簡単になるわけでもないのだが，全12面という遠大な大泥棒への道は，果てしないものがあったことは事実である。

そこでこの移植版なのだが，その難易度もきっちり移植されてしまっていることにプレイしてすぐに気づいたのは当然のことであった。

たしかにそっくりというのはうれしいのだが，難しいままというのも納得がいかず，複雑な心境に陥ってしまうのは人間のわがままのような気がする。

だからプレイしたとき，苦手だった場所で昔と同じように悲鳴をあげて倒れる主人公を見たときの気分は，大事なものと引き替えに前からほしかったものが手に入ったときに味わう，現実の厳しさの酸っぱい部分のような感じがして，ちょっぴり悔しい気がしたのである。

だが，完全移植といえるのはとてもうれしいことである。それだけX68000の機能が優れているわけだし，しかもその高精彩な

X68000用 5"2HD版2枚組　9,000円（税別）
シャープ　☎03(3260)1161

侵入前のブリーフィング？

練習モードもバッチリ

グラフィックの魅力が画面にあふれているのである。難しくてもいいじゃないか，何度でも継続できるのだし，歯ごたえのあるゲームを求める人もいるのだ。

そういわれてみれば，この「ボナンザブラザーズ」はクリアすることが直接の楽しみではないことに気づく。壁にはりつき警備の一瞬のスキを突いて駆け抜けるのが，このゲームの醍醐味であり，「わっはっはー」と高笑いをして悠々と「お宝」を手に入れる主人公の仕草が快感なのだ。

とすれば，きちんとそっくりに移植したことはほめられるべきであり，難易度も含めて，このゲームの完成度は高いということになる。これはゲームのスタイルを受け入れるプレイヤーの問題もあるので一概にいえないが，私はこの移植を結果的には満足のいくものと評価する。やはりいいものはいいのである。

## 泥棒よ大志を抱け◆◆◆◆◆◆◆◆◆◆◆

なにやらやたらと固い話になってしまったので，ゲームの中身のやわらかいところも紹介しておこう。簡単に各面を紹介しつつ，いくつかのトラップも説明したいと思う。どこで笑うかの種明かしのような気もするが，そこは勘弁してもらいたい。

**●1面　銀行**

特に何があるというわけではない。水色のガードマンはあとで強敵になるので，いじめておくと気分がいいかも。

プレス機で潰しても気絶するだけ

この俺様にハエが止まるとは情けない

しまった，気づかれたか

ボーナスステージ。叩かれまくり

ゲーム開始前のデモより

**●2面　大富豪の邸宅**

いきなりバナナの皮が落ちている。わざと踏んでもなんとかなる。ゴンドラで右の建物に移動するのだが，乗り損ねると真下に落下してしまうので注意。ただし落ちただけではミスにならない。

**●3面　カジノ**

屋上に登ったときに寝ている緑の警官は，落として消すことができる。トランポリンはあまり何度も跳ねていると逆に敵にやられるのでほどほどにしよう。

**●4面　造幣所**

もうここはプレス機にかぎる。敵を誘導してバコンバコン潰すのは最高。あまりにやりすぎての時間切れに注意？

**●5面　デパート**

最も楽しめる面。地下では果物を使って隠れられるが，3階の帽子売場のほうがオチャメで笑えるぞ。

**●6面　地下の金塊**

トロッコで登場と思いきや，いきなり行き止まりで投げ出されるのが情けない。微妙な段差が巧妙な面だ。

**●7面　宝石店**

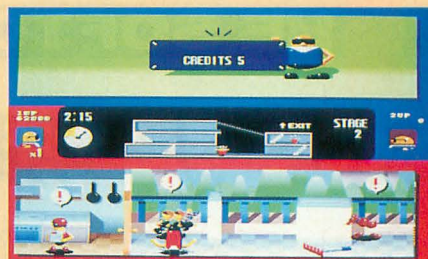
序盤が厳しい本格的な面。手前と奥のラインを使い分けよう。中盤のヤマである。

**●8面　研究所**

1階のロボットの頭を拝借する姿は爆笑間違いなし。それに，よく見るとロボットの姿が床に写り込みをしているのだ。

**●9面　豪華客船**

エンジン室に降りるとプレス機がある。テーブルの上のバナナは気づきにくいのでこれも注意。

**●10面　骨董屋**

トランポリンは左を全部集めてから使おう。ここらあたりからは敵も必死になってくる。無理に抜けるより，おちついていこう。

**●11面　美術館**

ゴンドラがある。降りると結構厳しいので手早く動くこと。屋上への階段の左から落ちるとかなり致命的なロスになるので試さないように。

**●12面　ピラミッド**

きわめて厳しい段差や迷路構造で，ターゲットの数も多い。最後に取ることになるターゲットのところには，落とし穴が2個あるので注意。

## 泥棒にうまいものなし◆◆◆◆◆◆◆◆

こうしてこのゲームを振り返ると，オリジナルの時点できめ細かく練られた演出や緻密な面構成，ゲームシステムが絡み合って，ひとつのゲームとして完成していると感じざるをえない。それを忠実に移植し，そのよく考えられたゲームがそのまま遊べることで，逆に今回「ボナンザブラザーズ」というゲームを見直すことができたような気がするのである。

X68000のアーケードの移植はいままでの状況を見ると，何かが犠牲にされても比較的許されてきた傾向がある。ところがそれも，「パロディウス」や，この「ボナンザブラザーズ」のような，オリジナルのコンセプトまでも忠実に移植された作品が現れたことで，どんどんユーザーの目が肥え，メーカーもそれに応える作品を送り出してくれるようになってきたといえるだろう。そういった意味からいってもこの作品の位置は，重要な意味を持つと思うのである。

### 泥棒の魂百までも

このゲームの設定では，敵は街を支配する悪者で，盗みに入る建物はみんな悪の温床という設定なのだが，「悪徳デパート」とか「悪の骨董屋」というのはどうもイメージがわかなくて納得ができない。どうせだったら「悪の出版社」とかあれば笑えるのだが，編集部の中を見渡してみても盗んで得になるようなものはないから，こういうネタは禁句だったか，うんうん。

| 総合評価 | 0　5　10 |
|---|---|
| ゲーム性 | ★★★★★★★ |
| グラフィック | ★★★★★★★★ |
| 移植性 | ★★★★★★★★★ |
| 演出 | ★★★★★★★★★ |
| CG度 | ★★★★★★★★★ |
| ボーナス失敗 | ★★★★★★★★★★ |

# ストーリー明瞭，されど波高し

Komura Satoshi
古村　聡

人気のRPG「ロードス島戦記」がいよいよX68000にも登場。最初は目的が明かされていませんが，仲間と出会ったり旅を続けていくうちに，次第に全貌が明らかになっていきます。フルマウスオペレーションなのもうれしいですね。

X68000にロードス島戦記が出る，というアナウンスが出てからどれほどの月日がたったのか。たいへん長らく，どころの騒ぎではない。もしかしたら，もう出ないのではないかと思ってしまった人もいたのではないだろうか。待っててよかったね。

さて，このロードス島戦記というゲーム，当然X68000用に発売されるのは初めてなわけでありますが，名前だけは聞いている方も相当いらっしゃるのではないかと思います。なにしろこのゲーム，同名の小説，オリジナルビデオなどがわんさと出ており，パソコンゲームとしても今回紹介する「灰色の魔女」に加え「福神漬」，そしてもうじき他機種で発売予定の「ロードス島戦記2 ～五色の魔竜～」と，メディアというメディアをまたにかけるマルチメディアヒット作だからなのです。私も他機種での評判はかねがね耳にしており，とても楽しみにしていました。

というわけで，我がOh!X編集室にピカピカのサンプル版ディスクがやっと届いたのであります。おそらくマスターアップ前アビキョウカン。プログラマもシナリオライターもデバッグの嵐のまっただなか，地獄のような忙しさのなかから，貴重な時間をさいて送ってくださったのでしょう。おかげさまで無事，レビューができます。

さて，ディスクユーティリティでロールプレイングゲームにお決まりのボーナスポイントの割り当てと職業を6人分決めると，いよいよゲームが始まります。

X68000用　5″2HD版3枚組　9,800円(税別)
ハミングバードソフト　☎06(315)8255

## 村をさまよう◆◆◆◆◆◆◆◆◆◆◆◆◆

ここはターバの村にあるホワイトドワーフ亭。数人の男がたむろする場末の小さな酒場。ここが冒険の始まりの地なのです。

「わしは北の村の出で，ドワーフ族の細工師ギムじゃ。わけあって旅に出ようと思っている。おまえさんも旅に出るのなら，わしと一緒にいかんか？」

はい，をクリック。

「なら友よ，一緒に飲もうではないか」

という感じでギムが仲間になります。では，酒場でいろいろな情報を聞いたあとで，今度は寺院へ向かってみましょう……。

「ここはマーファの寺院です。どなたかお困りですか？」

寺院を選ぶと，ちょっと小じわの目立つ司祭様のグラフィックが現れます。ロールプレイングゲームのご多分にもれず，この寺院は傷ついたり，毒を受けたものの治療や死体を生き返らせたりということができるのです。この死体を生き返らせることができるというのが，いかにもファンタジーですね。ここでは司祭の話を聞いてみることにしましょう。

「私が司祭のニースです。あなたたち，旅に出られるのなら，どうか私の娘レイリアを探してください。7年前に寺院に侵入してきた何者かにさらわれ行方知れずなのです。どうか，よろしくお願いします」

なるほど。これで旅の目的ができたわけですね。

ところで，このゲームは作ったキャラクターの名前を決めるのにキーボードを使う以外には，まず完璧にマウスひとつでゲームができるようになっています。俗にいうフルマウスオペレーションというやつですね。また村の中では酒場や市場，寺院といった場所への移動，ステータスや武器を装備するためのキャンプといったものは画面上にあるメニューをクリックすることでサブメニューが降りてきて，そのなかから選ぶ，いわゆるプルダウンメニューになっています。

さて，村を十分に歩き回ったところで，そろそろ外の世界へ出るとしましょうか。

## スクリーンは3モード◆◆◆◆◆◆◆

市場で武器を揃えてキャンプで武器を装備したら村を出ましょう。

村を出るとスクリーンはフィールド画面に移ります。地形は四角い，地形を表すチップで作られている，いわゆる典型的なウルティマ型のゲーム画面です。このゲームではフィールド画面上でもフルマウスオペレーションになっています。

ゲーム中にソフトウェアキーボードを出して，そーれ，フルマウスオペレーションだぁ，などという冗談ではありません。

画面上にマウスカーソルがあり，これを

この街から冒険が始まる

村長に頼まれたことを成し遂げて村に戻ると……

キャラの上下左右４方向いずれかにおき，ボタンを押すことでキャラクターが動きます。というか，このゲームではイースなどのように画面上をキャラクターが移動して，ある程度の位置まできたときに初めてスクロールするのではなく，なにがあっても背景のほうをスクロールさせて自分は常に画面の中心にいる，という方式をとっています。イースのように自分がちょこまかと動くようなゲームではないし(自分のキャラクターがちょこまかと左右に動くのであれば，イース式のほうが画面スクロールの回数は当然少なくなる)，第一X68000の場合スクロールはハードがやってくれるので１秒間に何回スクロールしようと関係ないのだから，この画面配置はX68000にとってはいい方法であるといえるでしょう。少なくとも私はこのほうが好きです。

さて，フィールド画面中を歩いていると敵と遭遇し，突然画面がフラッシュして戦闘モードに入ります。この戦闘モードにはタクティカルコンバットモードとクイックモードがあり，そのどちらかを選ぶことができます。タクティカルモードというのは，画面上のモンスターと自分のチビキャラがちゃんちゃんばらばらやる，シミュレーションゲームのようなモード。クイックモードというのは，画面中に文字だけが出てくるいわばウィザードリィのようなタイプになっています。

このタクティカルモード，見栄えはいいのですが，自キャラが敵を一匹倒すごとに，自動的に防御状態に入ってしまうのが欠点です。そのため，ほかの敵に向かうには，いちいち攻撃コマンドを選ばなければならず面倒臭いのです。私はいつもクイックモードを使うのですが，ゲームの節目になる大きな戦闘では必ずこちらを選択しなければならないようになっています。

画面のモードとしては，このほかに３Ｄ画面モードがあります。これはダンジョンなどに入ったときになるモードで，ウィザードリィ的な画面(ただし，線ではなく面で

城の中はやっぱりダンジョン

砂漠にはなにやら意味ありげなピラミッドが

描かれる)，操作法を思い浮かべればだいたい当たっています。

## 感想

このゲーム，なかなかイベントも豊富で，少なくとも飽きさせません。たとえば，序盤のゲーム展開にしても村でいろいろ聞いたあとで，フィールド画面で経験値を稼ぎ，地下のダンジョンに行って，都に行ってトーナメントで試合をして……と，最初は何をやったらいいのかと迷うこともあるかもしれませんが，だいたいは自分から動くようにすれば次から次へとやるべきことがあるので，無意味な経験値稼ぎに陥ることなく，わりと楽しく遊べます。

ただ，このロードス島戦記，よくやっているところも目立つのですが，半面，アラもよく目立つゲームなのです。

たとえば，グラフィック。洞窟などの中の３Ｄ画面形式のダンジョンを見てほしいのですが，このグラフイックはなかなかいけると思いませんか？ この前にハミングバードソフトからX68000に出た「ラプラスの魔」の３Ｄ画面に似ているのですが，壁やドアの微妙な光りの加減といい，壁のすみのちょっと汚れたような色加減といい，なかなか質感もあり，リアルな感じがして非常に雰囲気にあっているのです。

が，しかし，あれほどのグラフィックをゲームに使うだけの実力がありながら，村などの粗っぽいグラフィックはいったいなんなのでしょうか？ 最初の村の木のキリカブなどは妙にぺったりしていて，X68000のゲームとしては非常に恥ずかしいです。少なくともラプラスの魔のほうがかなりよく描けていたように思えます。

プログラムにしても，オープニングでラスタースクロールを使ったりと，けっこう技術力があることを見せつけてくれます。しかしその半面，まったく同じ文章を表示するのにもいちいち書き換えるらしく，そのためちらついたり時間がかかったりしてしまうのです。もう少しプログラムに無駄がないかチェックすればいいのに。

まぁ，これがハミングバードソフトとしてはまだ２作目でありますので，きっとX68000に慣れていないせいもあるのではないかと思いますが……。次回作ではぜひとも頑張っていただきたいと思います。

最後にちょっとした裏ワザ。実はこのゲーム，キーボードにもちょこっと対応していて，「はい」はテンキーの１かリターンキー，「いいえ」は０かESCキーで選択できるのです。ほかにもいくつか裏ワザがRead me. docに隠されていますので，興味のある方はのぞいてみてください。

さて，他機種ではこのロードス島戦記，２が出るのですよね。ということはX68000版もすぐに出るんでしょうか……。今度は待たせないでくださいね。期待してます。

中ボスとの戦闘はタクティカルモードで行われる

### ゲームとしては遊べると思う

本文ではかなりきつい書き方をしてしまったが，けっしてよくないゲームではない。いや，むしろ，ゲームの流れも快適で，遊べるという意味では，かなりおすすめできるゲームだと思う。

ただ，X68000用に発売されたロールプレイングゲームのなかではそこそこいいセンいっているだけに，システム/グラフィック面でのアラに腹が立つだけなのだ。技術的にはいいものを持っているだけに，ていねいに作ればもっといいものができたものを，そういう思いである。

また，本文中ではいわなかったが，シナリオに関する説明不足も少し目につく。たとえば，最初のほうの空き家で手紙を見つけ，城に行って格闘大会で優勝したあと，いきなり，王様に「ロードス島の南のほうの国の情報を調べてくれ」といわれてしまう。いったいあの手紙には何が書いてあったのか？

まぁ，いい。文句ばかりいっているようだが，私はいいと思っている，この１本である。

| 総合評価 | 0　　5　　10 |
|---|---|
| 操作性 | ★★★★★★★★ |
| ビジュアル | ★★★★★★ |
| サウンド | ★★★★★★ |
| テンポ | ★★★★★★★★ |
| 熱中度 | ★★★★★★★★ |

# 悪夢は再びディスプレイに

Nishina Takashi
仁科 隆司

**宇宙開発への基盤として，膨大な費用を投じて建造された汎用軌道基地ジーザス。そのジーザスから発進した2つの探査船が採集したハレー彗星の中には恐ろしい宇宙生物が潜んでいた。そして，4年……。**

いやあ，やられたな。はっは。

なにがって，この「ジーザスII」ですよ。想像以上に面白かったんで気分がいいナ。ボクは前作の「ジーザス」が結構好きだったんです。続編が出ると聞いた3秒後に，「レ，レビューをやらせてくださいっ」と頼みにいったぐらいですから。

前作の「ジーザス」は映画的手法を意識したアドベンチャーゲームでした。「さあ，どのへんでモンスターが出てくるのかな」と思ってやり始めると，いきなり人がモンスターにやられている。「おわぁ，何じゃこりゃ」とのけぞると，それが「変身モンスターゲーム」の画面だったというオチがつくわけです。画面のパンニングやアニメーションにも凝っているので，かなり「見せる」ゲームに仕上がっていました。

で，前作から経つこと4年。話によると，この4年がまるまる「ジーザスII」の制作期間だったそうです。4年間全力で開発ということはないだろうけど，とりあえず細かいところを詰める時間は十分にあったはずですよね。

## 恐怖と戦慄の鼓動 ◆◆◆◆◆◆◆◆◆◆

**シーン0**

西暦2046年，ハレー彗星の接近にともない人類初の有人探査計画，「ジーザス計画」が実行に移された。「ジーザス」とは銀河防衛軍所属の軌道上ステーションだ。「んーもう，JESUS」というのとは関係ない。

1号機コメットの乗員はハレーの氷片に潜んでいた地球外生命体に接触，乗員4名のうち3名が死亡。調査，救助にきた2号機乗員の武麻速雄はモンスターがコンテナにいることを確認，コンテナ自体を切り離すことでひとまず難を逃れた。

だが，モンスターは分裂し，おのおのが再生していた。モンスターはコンテナの中の1匹だけではなかったのだ。脱出した2人のシャトルに潜んでいたモンスターは2号機ころなにも現れ，さらに3名の犠牲者を出す。武麻速雄とエリーヌ・シュレマンの2人は死闘の末，あるメロディを使ってモンスターを快速艇に乗せ，太陽系外へ向かって発進させることに成功し，戦いにピリオドを打ったのだった。たしかにモンスターは去っていった。だが，コンテナに詰められたほうのモンスターは……？

**シーン1：貨客船カリスト号格納庫**

「おい，いま揺れなかったか，この船？」
「そりゃ，動いてるんだから多少は……」
「変だぞ和也，止まっちまってるぜ！」
「ホントだ，エンジンの音も聞こえなくなってる……？」

オレは五色和也。以前は銀河防衛軍の訓練生だったけど，いまは3WDビークルのメカニックをやっている。オレは親友の佐伯真治と一緒に，この貨客船カリスト号に乗っていた。5日後にモナコで開催される3WDビークルのレースに出場するためだ。

が，通信関係の故障で船は突然止まってしまった。故障の原因は誰かがケーブルを切ったためらしい。この船を故意に止めようとしているものがいる……？

**シーン21：第2客室**

麻世「でも，どうして私の名前を……？」
和也「仙さんに……，この船の航海士に聞いたんです。第2客室に，かわいい女の子が乗ってるって。アハッ」
麻世「ウフッ。五色さんていつもそうやって女の子に声をかけてるんでしょ」

必死の聞き込みにもかかわらず（ウソ度80%），犯人の手がかりはつかめない。

この船の乗員は4人。船長，松山機関士，航海士の仙さん，そして甲板員のイアルテ。乗客はオレたちを入れて5人。牧原麻世というかにもお嬢さんという感じの娘と，デザイナーのファーナ・アトキンスさん，そして医者の三井先生だ。誰にも船を止める理由がない。ただ，停船していたところに宇宙からコンテナが落ちてきたという事件があったが，あれになにか関係があるのだろうか？　悩んでいるところにファーナさんから船内電話が入る。

**シーン46：第3客室**

ファーナ「見たでしょ，あのコンテナ？」
和也「ああ，たしかコメットって書いてあったけど」
ファーナ「まず，私の身分から話しておくわネ。私がデザイナーっていうのは嘘なの。本当は軌道間貿易管理機構の諜報員，俗にいう通関Gメンよ」

そして，オレはこの船の正体を知った。
和也「でも，なぜ軍のコンテナを？」
ファーナ「4年前のハレー探査計画に地球外生命体が絡んでいたという話は知ってる？」
和也「まさか……。ゴシップ誌のガセネタだろう？」
ファーナ「ところが本当なの。コメットところなの空調システムが故障したという防

X68000用 5"2HD版5枚組　8,800円(税別)
エニックス　☎03(5272)2374

右が主人公の和也，左は親友の真治

衛軍の発表のほうがウソだったのヨ」
ファーナ「しかも，そのモンスターは人の遺伝子情報を読み取り，ごく短時間に進化を遂げるらしいのよ」
和也「じゃあ，あのコンテナの中にモンスターがいる可能性があるのか？　冗談じゃないゼ，すぐに防衛軍に知らせなきゃ！」
数分後，銀河防衛軍は次のような通信を傍受し，部隊の出動を決定した。
“コメットノ……モンスターガ……コンテナ……”

**シーン61：カリスト号キャビン**

船長が，松山さんが，そして仙さんが死んだ。そしてオレとファーナさんも，銀河防衛軍の到着が1歩遅かったらやられていただろう。コンテナの中のモンスターは死んでいた。なのに，なぜ再びモンスターは現れたのだろうか？

防衛軍のノーマン軍医が口を開く。
「個体維持が不可能になった生物が考えることはひとつだ」
「……子孫を残す？」
「そう，ヤツは動物的というよりは，むしろ植物的に子孫を残そうとしたのだろう。種子，あるいは胞子の形で。そして，それに感染すると，船長のように……」
すると，まだこの中の誰かがモンスターに“感染”している可能性があるのか？
そのとき，麻世ちゃんの悲鳴が！
麻世ちゃんの部屋に飛び込んだオレは，自分の目を疑った!!

## スピード・スリル・サスペンス◆◆◆◆◆

シナリオはここまででまだ半分も来てません。しかも，面白いのは実はここからなのサ。モンスターに接触した疑いのある人物はジーザスに隔離されます。誰かが実はモンスターかもしれない。緊張感が高まるなか，ジーザスの通信網がなにものかに破壊され，ジーザスは宇宙の孤島と化すのです。そして，エイリアンと人間とがジーザスのコンピュータの支配権をめぐって壮絶な戦いを繰り広げます。

牧原麻世，この娘が実は××の×……

和也を襲うモンスター。反撃の手段は？

前作のフォジーも進化して再登場

後半は次から次へとピンチが和也を襲う。もう「ダイ・ハード」も顔負けの息もつかせぬ展開。あとからよく考えると，考証は結構いい加減なんだけど，プレイしてる最中にはそれを感じさせないパワーがあります。SFサスペンスの常套手段を巧みにおりまぜて，プレイする人をグイグイ引き込んでくれますゾ。

ボクは1泊2日で観終わった（というのが正しいと思う）けど，2日目は4時間以上ディスプレイの前に釘づけ。映画じゃないんだから途中で席を立ってもかまわないのに，ストーリーの流れが切れるのがイヤで離れられない。もともとボクはこのテの話が好きなので，コロッとまいってしまいましたゼ。

それから，文章がすごくよくできています。プレイヤーの次の行動をうまく導いてくれるので，話の流れがとぎれないのです。初めの部分もプレイヤーに状況を把握させようと気を使ってあるし，ゲーム中はいつも謎を抱えて話が進んでいく。このへんはさすがに4年かかってるだけのことはあります。本当にこのシナリオはスゴい！

しかし……，いいたかないけど画面はさびしすぎる。ボクは編集室の21インチディスプレイでやったけど，画面の迫力はそれでやっとまあまあという程度。ボク自身は「X68000でやる以上，絶対にグラフィックや音楽はパワーアップさせろ！」とは思わないけど，さすがにこれでは，ね。

デジタル8色で描いた絵はフルカラーにしてほしいけど，制作の都合もあるだろうしキレイだからまあ許そう。絵のサイズが小さいのもホントはすごくイヤだけど，演出との絡みもあるし。い，いいとしようか。だけど，ここまで譲ったとしても，この真っ黒なバックだけは納得いかないゾ。絵のサイズは「サイレントメビウス」と大差ないのに，見た目の豪華さが違いすぎる。やはり，PC-8801でオリジナルを開発するというのはもう限界なのでは……。

ただ演出はあいかわらずうまいです。サウンドエフェクトとBGMも映画音楽のようにいい働きをしています。これがすぎやまこういち氏のうまさってヤツでしょうか。要所で前作からのBGMをうまくアレンジして，引っ張ってきてたりしてネ。

ジーザスIIでは文章表示も速くなったし2度目3度目に表示されるときはパッと出てくるから，そんなにかったるくはありません。ゲームの間を大切にしたいという制作者の気持ちを汲んであげましょう。

ああ，なんかひさびさに元祖「ジーザス」もプレイしたくなっちゃった。今日はX1 turboを立ち上げてみようかナ。

### 一流のエンタテイメントだからこそ

アドベンチャーゲームというと多少なりとも，「プレイヤーに行動の自由を与えよう」という考えが浮かんでしまうものです。が，「ジーザスII」は思い切ってプレイを制限するところは制し，プレイヤーの行動を文章そのほかでうまく導くことによって，ストーリーのテンポと臨場感を高めることができた成功例だと思います。アドベンチャーというとすぐに「黄金の羅針盤」のような推理ものを思い浮かべてしまいがちですが，こういったエンタテイメント志向の作品もなかなかどうして楽しめます。

ただ，ここまでエンタテイメント志向であるからこそ，画面の迫力のなさが目についてしまいますネ。これがX68000で開発していたらみんな目をむくようなゲームになっていたかもしれないのに。ああ，もったいない。

まあ，この画面のさびしさが許せない人には，なんでこんなにボクが入れ込んでいるのかわからないかもしれないけど，一度頭の中を8ビット時代に戻してプレイしてみれば，この「ジーザスII」のよさがわかるハズ。激しくおススメします。

**総合評価**（0　5　10）

| 項目 | 評価 |
|---|---|
| エンタテイメント度 | ★★★★★★★★★★ |
| シナリオ | ★★★★★★★★★★ |
| グラフィック | ★★★★★★ |
| サウンド | ★★★★★★★★★ |
| 熱中度 | ★★★★★★★★★★ |
| X68000らしさ | ★★★ |

# 富国強兵だよ，えんやこらさ

昨年発売されたシミュレーションゲーム「シュヴァルツシルト」の続編。プレイヤーはオーラクルムの王となり，国を安全かつ強大にしていくことが目的だ。もちろんフルマウスオペレーションだ。

Yamato Satoshi
大和　哲

どうも，はじめまして。と，典型的な日本人のあいさつをしてしまいました私は，大和哲と申します。以後，ごひいきに。

さて，この工画堂スタジオというソフトハウス，かつて8ビット機の時代，コズミックソルジャーなるゲームを出していたことで覚えておられる方もいらっしゃることでしょう。かくいう私もプレイしましたが，いやいや，あれはなかなか楽しいゲームでありました。このコズミックソルジャーは，当時としても非常に珍しかったSFロールプレイングゲームで，敵をただ倒すだけでなく半殺しにしておいて金や情報を奪ったり，画面の1/4をハイレグのねーちゃんが占めていたりしまして。当時まだ若かった私は，広告を見てこいつが主人公かと思って，異常な興奮をおぼえてしまった(おひおひ)という，なかなかユニークな作品だったのであります。

さて，今回紹介する，このシュヴァルツシルトII・帝國ノ背信というゲームですが，大宇宙シュヴァルツシルトを舞台としたものになっており，またシナリオシミュレーションゲームという一風変わったジャンルのゲームを名乗っています。はてさてその出来やいかに。

それでは，さっそくマウスで実戦モードをクリックして（このゲームは実戦モードと演習モードに分かれている）ゲームスタート。あなたの目はあなたの体を離れ，この，遥かなるシュヴァルツシルト大銀河へと旅立っていくのです。

X68000用　5"2HD版2枚組　9,800円(税別)
工画堂スタジオ　☎03(3353)7724

## 吾輩は宇宙をうれう皇帝なり◆◆◆◆◆

巨大な渦状を呈し，誇らしげに光り輝く大銀河シュヴァルツシルト。幾百億の星々が数百億の歴史を持ち，幾千億の人類を育んでいる。しかし，いま人類は新たな危機を迎えようとしていた。そしてその兆しは，ここソマリ星団において顕著な形として姿を見せ始めた。時に星暦3964年……。

どーもどーも。私がオーラクルム皇帝でありんす，へっへっ。私の率いるオーラクルム星国（星がいくつか集まってできた国だから星国なのだ）はいま，だーいぴーんち！なのです。オーラクルムは三方を小惑星帯に囲われ，唯一の隣国トリスティアが我々の西に位置しているという地形。

しかし，最近この隣国トリスティアが，そのまた隣国であるロッサリアの本格的な武力侵攻って……簡単にいうと“ねーちゃん痛い目にあいたくなきゃ金だしな”なんだけど，そーゆーわけでいま，滅亡のガケっぷちに立たされているわけだ。もともと我がオーラクルムは隣国のトリスティアとは旧来より仲のいい国だし，現在では攻守同盟を結んでるから，どちらかが攻め込まれたら協力してこれを退治しなきゃいけないんだけど……。

あんたねぇ，攻め込んできたロッサリアの背後には，さらに強力なゲルーマンとかグレイブリーとかいう軍事国家が後押ししているって噂なんだよ。そりゃぁね，この広い銀河の中にはそれらをよく思わない国もあるし，それらに協力を得られればいいけど……。いまのところそういう予定は全然なし。まさに，四面楚歌。どーしろっつーの。だいたい私んとこは，いま，軍艦も工場も足りないんだからね。

で，皇帝閣下何かご命令を，だって？　そーだね。それじゃぁ，まず，小惑星を探査させようか。もし，大判小判がざっくざくの小惑星が見つかったら，わしゃ世界一の大金持ちだぜ。宮殿に美女をはべらせてぇー。じゅるっ。……おっ，なに？さっそく見つかった⁉　よし，ハーレムだハーレム！　なに，そんな大きなもんじゃない？せいぜい工場で使うようなものしかでないって？うーん，しょうがないな。

それじゃぁ，富国強兵作戦だ。造船所を作れ，船作れ。たくさん，軍艦を作ってよその国を攻め取って大金持ちぢゃぁ(悪魔かお前は)！

ピーッピーッ！　おや，トリスティア最高政府より通信が入ったのね。

「宇宙座標0903上にて，我ロッサリア軍と戦闘状態に入れり。至急，貴国艦隊の援軍を願いたし」

むむーん。援軍たってねぇ。こっちが援軍出したって，手持ちは32艦だぜ。合わせてもオーラクルム・トリスティア連合軍は総勢137。対するロッサリア軍は……げ，240もあるじゃない。大軍だよこりゃ。うーむ，もっと船作って強い軍隊にしなくちゃ対抗できないなぁ……。よし，援軍は出さない！トリスティア星国さん，私のために犠牲になってね（おーいっ！）。

## 敗戦必死！　滅亡への秒読み◆◆◆◆

というわけで，あわれトリスティアは滅んでしまったのであります。トリスティア星国さん，君のことは忘れないよ。さあて，それでは我が国は富国強兵の道をひたすら

トリアスティアから援軍の要請が。どうする？

歩むとしよう。さぁ，もっと造船所を作れー，軍艦作れー。びしびし。むふふふふ。かなり宇宙艦隊も大きくなってきたな。

ピーッピーッ。おや？　なんとロッサリアからの通信だ。トリスティアが滅亡して間もないのにご苦労なことだね。

「オーラクルム皇帝に告ぐ。

我がロッサリアに対し，敵対することのなきよう，ここに貴国摩下の全艦隊を放棄するよう希望する

ロッサリア最高政府」

へっへっ。ホールドアップですかぁ。こっちはもう軍艦の数では負けないもんねー，そんなオドシにのらないよん。おーい，通信兵。返事はな，“バカめ”と送ってやれ。え，“バカめ”だよ，ワカメじゃねえって。

おーっと，さっそく宣戦布告してきなさったね。よーし，我が精鋭部隊よ返り討ちにしてやれ！　こちらの惑星のまわりに味方の迎撃機。向こうに敵の爆撃機と支援戦闘機。よーっし，撃て撃て！

……あ，あれ？　えーっ？　なんじゃこりゃぁ！　こっちが一方的に負けてるじゃん！……ああっ，軍艦の数は多いのに能力が全然ない！　そっか，ちっとも模擬演習で訓練させなかったからシロートばっかりの軍隊になっちゃったのか！　うっそぉ。がんがんやられてく……。あああ，味方の第2，3，4艦隊が全滅して，戦場になった惑星

各国が持つ戦艦の一覧。武力差がひと目でわかる

造船も大切だけど，模擬演習もやらないと……

ロッサリアに宣戦布告されてしまったぞ

ナッソスは敵の手に落ちてしまった。あと残った戦艦は第1艦隊の4艦だけ……。ぼーぜん。ああ，敵が攻めてくるよー。あのときトリスティアを助けとくんだったーっ！　だれか助けて，神様ー っ！

## 進行は楽で楽しい◆◆◆◆◆◆◆◆◆◆

だがしかし！　なんとオーラクルム星国はほとんど瀕死のところで助かってしまうのですね。どうして助かったかは，ま，プレイして確かめてくださいな。

このゲームは，シナリオシミュレーションというその名のとおり，なかなかいろいろなイベントが起こります。こちらがある国に対して戦争を起こそうとすると，別のある国が「その国を滅ぼそうとするやつは承知しない」と戦争をしかけてきて，地域戦争が泥沼の全面戦争になってしまったり，あるいは敵に攻められて負けそうなときに，いきなり味方してくれる国があったりとなかなか波乱万丈に満ちています。

半面，戦いに勝ちさえすればどんなストーリーの進め方をしようと思っても結局は同じような道をたどる，悪いいい方をすれば一本道的で，このゲームはシミュレーションゲームというよりはむしろイースのようなロールプレイングゲーム的(無論これらのゲームも本来の意味でのロールプレイングとはかなりかけ離れているようだけど)な要素が多いゲームといえそうです。

またこのゲームは，シミュレーションゲームとしてはコマンドもゲーム進行にかかわる要素も比較的少ないです。メインのコマンドウィンドウは艦隊，民事，など7つのコマンドに分かれ，このコマンドの下にまたサブコマンドがいくつもあるのですが，本当に必要なコマンドはそう多くはないのです。基本的には富国強兵という目標を持っていて，豊かであるために所有する星を増やし，強くあるために資材を軍艦に変える。と，それさえ押さえていれば非常に楽で楽しいゲーム展開となるゲームなのです。

## 個人的な感想◆◆◆◆◆◆◆◆◆◆◆◆◆

このゲーム，移植自体は非常によくできていると思います。操作はフルマウスオペレーション，グラフィックも512×512の256色モードを使っているようで非常にきれいですし，文字表示や絵の出力をさせるようなコマンドを使用してもスピード的にも問題ありません。ただ，原作にかかわることで，X68000版の問題ではないのですが……。まだまだ，このゲーム，“シナリオ”ゲームを名乗るにはまだ中途半端です。

たとえば，人物像というものが見えてこないのはなぜなのでしょうか？　このゲームは百数人の光の勇者が始皇帝のもとに集い宇宙に平和をもたらす，という大絵巻の一部である，とされているのに，ゲーム中，いまいちキャラクターが見えてこないのです。彼らはただ，「利益を守るために宣戦布告」し，攻守同盟を結べば「平和のために非常に喜ばしい」ので受諾する。生きているキャラクターに見えてこないのです。どうせ，“シナリオシミュレーション”を名乗るなら，その辺もちょっと頑張ってほしかった，というのが私の意見です。

とはいえ，そんな偏ったマニアのたわごとにさえ耳をかさなければなかなか楽しく遊ばせてくれるゲームではあります。秋の夜長は，星を眺めながら，こんな楽しいゲームで宇宙に思いをはせてみませんか？

### 隣と違うこのゲーム

なぜか，SF風のシミュレーションゲームは発売日がかたまる傾向にあるようで，前作のシュヴァルツシルトも銀英伝と同じ号にレビューだったし，今回もなんでも同じく銀英伝とインペリアルフォースが発売日が近いようで，なにやら，まさかわざとやっているわけでもあるまいな，と思ってしまいます。

私はインペリアルフォースも銀英伝もやったことがないので最近はやりの比較広告するわけにはいきませんが，シュヴァルツシルトは2作ともいいゲームだと思います。なんといっても，適当にゲームをやっていてもシナリオという形でアドバイスやインフォメーションがあるというのは，初心者に限らず誰にとっても親切であり非常によいことだと思います。また，システムのレスポンスもスパスパとキレがよく，プログラムの育ちのよさを思わせてくれます。おしむらくは小さくまとまりすぎていて，少し爽快感に欠けることなのでしょうが……。

もちろん次はやってくれるはずです。頑張れ，工画堂スタジオ！

| 総合評価 | 0　5　10 |
|---|---|
| グラフィック | ★★★★★★★★ |
| 音楽 | ★★★★★★★ |
| シナリオ | ★★★★★★ |
| システム | ★★★★★★★★★ |
| おすすめ度 | ★★★★★★★★ |

# 銀河の歴史をもう1ページ

Kaneko Shunichi
金子 俊一

小説やアニメから人気が飛び火して，パソコンゲームになったSF大河ドラマ「銀河英雄伝説」。このゲームは続編やシナリオ集によってバージョンアップされてきたが，この「DX+kit」の発売でさらなるパワーアップを遂げる。

システムのよさを売りにした銀河英雄伝説IIから約1年，ついに続編ともいうべき銀河英雄伝説IIDX+が発売された。銀英伝，銀英伝パワーアップ&シナリオ集，銀英伝IIと続いたシリーズ第4弾にあたる。

基本的には銀英伝のシステムをそのまま受け継いでいて，ユーザーにストレスを感じさせないゲームに仕上がっている。

## スゴスバの銀英伝◆◆◆◆◆◆◆◆◆◆

銀英伝はシミュレーションゲームとしてはめずらしく操作性に優れている。マウスひとつでの艦隊運用もイメージどおりにできるのだ。レスポンスもいいので，マルチウィンドウのソフトをいじっているような感覚さえある。全体的に統一された操作方法は無意識のうちにやりたいことができ，扱いやすいことこのうえない。

また，HEXにこだわらず座標系だけで表すという表現方法もパソコンならではて，これからのシミュレーションゲームのひとつの方向性を明確に示しているだろう。

## モナコじゃないけど税金はない◆◆◆

銀英伝には税金制度があり，内政にも重点が置かれていた。この制度は面白かったと思う。銀英伝IIになってこの制度は廃止されてしまった。最初から艦隊を惑星に振り分けて，その中から出撃や増援をするように変更された。惑星ごとの治安や反乱といったパラメータはほとんど無意味になってしまっている。もったいない。いい方を変えれば，それだけ戦略に専念できるのだろうが，やはり内政にも気を使わないようでは，真の指導者とはいえないだろう。

これ以外にも銀英伝シリーズにはそれぞれ4者4様に細かい違いがあるが，必ずしもあとから発売されたものが優れているとはいえないようだ。もちろんトータル的にはよいほうに向かっていると思えるのだが。

## 違いがわかる男になれるか◆◆◆◆◆

拡張ルールでは難易度の設定が3段階選べる。いままでなかったのが不思議なくらいだが，ユーザーへの配慮がうかがえる。

また，提督の艦隊が全滅したときに脱出するかしないか選べるようになった。敵の提督は憎くとも，大切な提督は死んでほしくないもの。でも，必ずしも脱出できるとはかぎらないようだ。

敵のユニットと衝突したとき通り抜けるか抜けないかの設定もできる。ありがちなHEX戦のゲームでは敵のユニットはおろか味方のユニットまで通り抜けられないもののもあるようだが，いままでの銀英伝ではたとえ敵のユニットでも通り抜けができたのだ。やはり舞台は宇宙といえど敵のユニットは通り抜けられたらおかしいと感じる人もいるのだろう。私はそうは思わない。

艦隊の初期構成も変更できる

艦隊や提督は任意の惑星に配備することができる。さらに，提督には指揮下に数ユニットの艦隊がいるのだが，初期構成を選んだり，フォーメーションを変更できる。キャンペーンモードではここでの設定を覚えてくれているようだ。ただし，BGMはシナリオごとにボレロに戻ってしまうぞ。

キャンペーンモードでは勝敗にかかわらず，先のシナリオに進めるようになった。これは大きいだろう。どうしても先に進めずに挫折した人には福音だろう。各提督はレベルアップしやすくなったし。

目に見えない改良点としては，思考ルーチンがよくなっているように思える。結構面白い攻め方をしてくるので，適当にやっているとびっくりするときがある。艦隊の向きが重要で後ろから攻めるともろいのだが，なかなか後ろはとらせてくれない。時間差はさみうち攻撃などをすると素直なコンピュータは後ろを向いてくれるけど。

作戦中の艦隊の航路が表示される

X68000用 5"2HD版3枚組　5,000円(税別)
ボーステック　☎03(3708)4711

### わんぱくでもいい，たくましく育ってほしい

シナリオは全部で10本あり，イゼルローン要塞をヤン・ウェンリーが無血占拠をしたあとから始まる。原作では中盤にさしかかったあたりだろう。シナリオ別に登場できる提督を限定したほうが原作に忠実になったかもしれない。

今回で3回目のマイナーチェンジになる。いずれも変更は細かいルールや設定，機能などにとどまっている。最初のシステムが秀逸だったため，無意味に基本的なシステムやゲームの流れを変更しなかったことは評価に値する。ただし舞台が宇宙のため，マップやシナリオごとの特徴が薄くなってしまい，4本目ともなるとさすがに新鮮みに欠けてしまう。

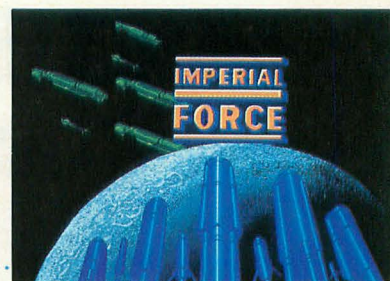
シナリオの自由度が少ないのでむずかしいのかもしれないが，もっと内政に力を入れたものなど，全然別のシステムでも遊んでみたい。

**総合評価**（0〜10）

| 項目 | 評価 |
|---|---|
| システム | ★★★★★★★★★ |
| グラフィック | ★★★★★★★★ |
| サウンド | ★★★★★★ |
| 新鮮み | ★★★ |
| 戦いのあとの紅茶 | ★★★★★★★★ |

# 箱庭宇宙の王者になろう

システムソフトといえば「大戦略」という感じだが，最近は同じシミュレーションでも，ちょっと変わった背景を持つものにも挑戦している。今回移植された「インペリアルフォース」もそのひとつで，宇宙を舞台としたゲームだ。

Urakawa Hiroyuki
浦川 博之

「インペリアルフォース」というのは英語で「帝国軍」という意味だ。帝国軍というと，「黒いオヤジがコーホーいってるワルの集団！」とか，「美形の金髪がマントをひるがえす耽美な集団！」とか，人によっていろいろ連想するモノがあるかもしれない。あるかもしれないが，この「インペリアルフォース」はハゲのオヤジやら頭のめりこんだ怪人たちが，宇宙の覇権をめぐってわさわさと戦うというゲームである。「我が征くは星の大海」というのは同じだが，ずいぶん似合わない気もするな。

## ラクに覚えてサクッと遊ぼう◆◆◆◆◆

ヘビーなシミュレーションゲームをプレイしている人だって，最初は元祖大戦略や雑誌に載ってる読者投稿のゲームから始めたはず。しかし，いまシミュレーションゲームに挑もうとしても，「ほお，こういうもんなのか」と簡単に試せるソフトはなかなかないのが現状だ。

そこいくと，この「インペリアルフォース」はシンプル。生産，戦闘，占領だけでゲームが構成されているし，しかも戦闘は自動にもできてしまう。だから，ゲームが始まったとたんにキーボードの前で石になってしまうなんてことはない。「ダッシュ野郎」のシミュレーション版といえるかもしれないな。あ，シンプルといえばエルフの「FOXY」なんてのもあったか。でも，「インペリアルフォース」のほうがシステムソフトが作っている分だけバランスがいいな，やはり。エッチな絵はないけど（まともに比べるなっつーの）。

ちょっとクセがあるとしたら，このゲームがリアルタイム方式だという点かもしれない。艦隊や惑星に指示を出し終わったら時計をどんどん進める。任務を遂行し終わったら知らせてくるから，時計を止めて次の行動を教える。そういうシステムだ。

宇宙船には大ざっぱにいって５つの種類がある。強いけど作るのに時間のかかる戦艦，あんまり強くないけど早く作れる巡洋艦，とにかく足の速い高速艦，未知の空間を調べてくれる偵察艦，それから星を占領するための強襲艦だ。初めの頃は強襲艦と偵察艦を順番に作り，星を占領して生産能力が高まってきたら，巡洋艦なんかを作るのがセオリーだね。

それから星系同士が近いと水色の線で結ばれる。これはネットワークといって，生産力のつながりを示すものなんだけれど，同時に物質の転送も行えるのだ。これが結構ポイントで，A星で作った戦艦をB星に転送してそこから発進させたりできるわけ。遠いところまで艦隊を移動させるよりは，こっちを使ったほうが速い場合もある。

凝ったシステムといえばホントにこれくらい。これだけわかれば宇宙船をガンガン作って，未知の領域をどかどか開拓して，星があったらぼこぼこ占領して，敵艦隊に会ったら逃げるか戦うかする。すべての星はキミのものだ。がんばれー。

## もっとサクサクと遊びたいぞ◆◆◆◆◆◆

さて，たしかにシンプルなゲームというコンセプトはいいけれど，苦言を呈したいところはいくつかある。

まずゲームシステムはシンプルで軽快なのに，プログラムが「軽快」じゃないこと。ディスクアクセスが多いし，画面の切り替えも時間がかかるほうだ（まだ，サンプル版なんだけどね）。ゲームをシンプルでサクサク進むものにしたいんなら，処理でもそれなりにがんばってくれないと……。

それから，これはどのシミュレーションにもいえるんだけど，ほしい情報がほしいかたちで手に入らないことが多い。ルールはシンプルでも，操作するときにとまどうようでは，説得力は半減してしまう。このゲームでは宇宙船がいつ完成したのかがよくわからない。完成するたびに報告があるべきだと思う（ひょっとして，後半になるととてつもなくうざったくなるかもしれないが）。こういうタイプのゲームでは「おう，ハチ！」「へい，がってんだ。親分！」というレスポンスのよさが不可欠だが，残念ながらそこまで到達していないようだ。

PC-9801版に比べてグラフィックなどはがんばって作ってあるのだが，コンセプトと実際の操作感覚が一致してないのが今後に残された課題だと思うぞ。

まずは偵察艦などを作って様子を探る

X68000用 5"2HD版2枚組　8,800円(税別)
システムソフト　☎092(752)5278

### シミュレーションの裾野を広げる作品だ

いちばん初めの「大戦略」を宇宙版にしたようなゲーム。いかに少ない要素だけで楽しめるものにするかをよく考えてある。ここらへんはSLGメーカーの雄，システムソフトの貫禄を感じる。なんでもかんでも詰め込めば面白くなるもんじゃないという考え方には賛成だ。

とはいえ，それなりにセールスポイントが必要なのはたしか。「インペリアルフォース」の場合は，ライトなルールに見合った軽快な操作性であるべきだったと思うんだが。ライトなシミュレーションゲームがやりたい人にはオススメできるだろう。

| 総合評価 | 0　5　10 |
|---|---|
| 操作性 | ★★★★★★★ |
| グラフィック | ★★★★★★★ |
| 音楽 | ★★★★★ |
| 入門ソフト度 | ★★★★★★★★ |
| 熱中度 | ★★★★★★★ |

# とりあえず，優勝してみました

Ogikubo Kei
荻窪 圭

荻窪圭はやりました。忙しいのにやりました。……そして，見事に首位。まあ，コンピュータはそんなに強くはないけれど。ちなみに記事中の球団名や個人名は著者の気分の問題で実際のゲームに登場するものとは違います。

ああ，よかった。先月号発売時点ではまだ中日が首位だった。まったく信じていなかっただけに，ほっとしている。

さて，なにはともあれ，戦力を分析してみるか。とまあ，各チームを覗くわけである。とりあえず，ドラゴンズである。落合が化け物だ……。西本のシュートがよく曲がる，与田のストレートが滅法速い。……それだけである。足がそこそこ速いのは立浪だけだし，守備がいいのも立浪だけだ。あ，そうか。このゲームはあまり極端な設定じゃないんだ……。

ところが，である。

ほかのチームを見たら，凄いのだ。槙原は球が速いうえに変化球もよく曲がるし，野茂に至ってはフォークが5である。ドラゴンズなんて西本のシュートが3なだけで，ほかの投手は皆1か2ばかりなのだ。どうして，北別府が150kmなんていう球を投げられるのだ。打者だってそうである。ドラゴンズの場合，すべてのパラメータが落合の打撃に集中したかのようだ。

こんなことがあっていいものか。ああ，こうなると，既存チームの変更がまったくできないのも善し悪しである。畜生，優勝してやる。

## 目指せ,全勝優勝!◆◆◆◆◆◆◆◆◆◆

そこで，ペナントレースへと突入する。難度は普通。リーグは現実のとおり。試合数だけは1カ月で1シーズン分戦え，という編集の意向により，なんとかできそうな65試合（つまり，半分だけですな）に設定した。

いよいよ開幕。対戦カードは勝手に決められ，同一カードで，2連戦か3連戦となる。

結論からいおう。大洋はずるい。なんといっても，足が速いのである。それでもって，前進守備やらあらかじめ守備位置をシフトしておくとかがまったくできないから，大洋に落ちる球や球威のある球は禁物である。ぼてぼてのゴロは内野安打になり，高めの球で打ち上げさせると，内野と外野の間にポテンと落ちる。

とにかく，低めに球を集めて，地道に，内野安打の2つや3つに腐らず，アウトをとっていくしかない。守備力を鍛えられる1日であった。結果はもちろん，落合のホームラン3発を含む大勝。

ポイントは2つだ。ピッチャーは確実に低めぎりぎりに球を集め，フライを打たせない。そうすれば，よほど内野手の守備力が悪くないかぎり，0点から1，2点で抑えられる。フィルダースチョイスでランナーをためなければ，たいていは完封だ。

もうひとつは，ポイントゲッターの前に確実にランナーをためることである。これだけできれば，まず負けない。写真を見ればわかるように，ゆうゆうの開幕21連勝である。

ここで，私は東京を1週間ほど留守にするため，編集のA氏に生中継68を託す。21連勝していれば，多少のことがあっても，優勝を逃すことはないだろう。

*　　*　　*

荻窪圭，夏の北海道へと飛ぶ。半分は仕事で半分は遊びだ。

北海道といえば，札幌。札幌といえばZOOMである。意味もなく，ZOOMへ遊び

ホームラン。きれいな夕焼けに花火が

そして，アウトを1つひとつとっていく

若狭湾 CENTRAL VS 恐山 PACIFIC 若狭湾スタジアム

X68000用　5"2HD版2枚組　9,800円(税別)
コナミ　☎03(3264)5678

### 日本シリーズ速報

日本シリーズくらいちゃんとやろうということで，私と，バッファローズの優勝を信じていたのに裏切られたA氏との間でドラゴンズーライオンズの日本シリーズが始まった。結果は4勝2敗でライオンズの勝ち。

うーん。やはり人間同士が一番面白いというのは生中継でも同じであった。そういうものかね。

このゲームも，オリオンズのレフトは肩が弱くてホームまで球が投げられないとか，内野フライでもランナーは走ってしまうとか，同様の理由でタッチアップができないとか，解説者のコメントが情けないとか，打撃10傑や防御率10傑を見せてくれないとか，BGMが凝っているわりにつまらないとか，コンピュータ同士の試合がかなりいい加減（100試合分くらいデータを持っておいて，それをランダムで引き出すくらいでもいいと思う）だとかいろいろあるが，総じてノリがいいのでとりあえずは許そう。

「生中継68 '92年版」に大いに期待だ。

にいき，蟹を食べさせてもらう。ついでに，開発途中のジェノサイド2をちょっとだけ見せてもらう。へへへ。

念願のフゴッペ洞窟へ行く。続いて手宮洞窟も回りたかったのだが，手宮洞窟が工事中で入れない。しかたがないので，近くの「喫茶 古代文字」でお茶を飲む。うーむ，残念。

富良野へ行く。中島朋子を探し回るが，どこにもいない。おかげで疲れる。

観光地化の激しい北海道をあとにし，やがて，帰京。

＊　＊　＊

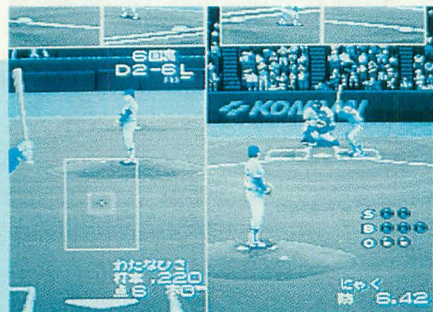
ああ，編集のA氏が関西出身だというのをすっかり忘れていた私が悪かった。A氏から返してもらった生中継68はすっかり様変わり。タイガースが僅差の2位につけているではないか。

とにかく，ペナント再開である。

打線は水物。これだけは正しい。15点も取ることがあれば，1点しか取れないこともある。力のないやつがラッキーなポテンヒットを打つこともあれば，そこそこ打てそうなやつが，打球が伸びすぎるせいでみんな外野手に取られてしまうケースもある。「あ，抜けた！」と思った当たりが取られてしまうのは非常に悔しい。

やはり信頼できるのは，立浪のバントヒットと，落合のホームランだけであった。ナイターでホームランを打つと，写真のように花火が上がる。これが楽しみである。難度が普通になっていると，相手のピッチャーがなかなか甘い球をよく放ってくれるので，うれしい。ウイニングショットの鋭い変化球をばかばか投げられた日には，点なんてそうそう取れるものではないのだ。

雰囲気は非常によく出ているだけに，前進守備や，右寄り/左寄りといった状況に応じた守備位置の変更ができないのは残念だったね。

ついでに走塁。フライが上がったら，2アウトじゃないかぎりスタートを切らない，とか，ハーフウェイで待つとか，してくれよ。でもちゃくちゃくとペナントレースは進む。もう，タイガースも遥か彼方だ。セ・リーグはもはやぼろぼろで，5割に達しているのはドラゴンズとタイガースだけ，という有り様だ。

日本シリーズの息詰まる戦い

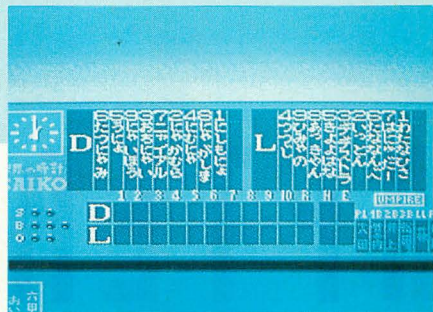
ラッキーセブン。よりいっそうのご声援を

涙で見えない胴上げシーン(ウソ)

ハイパースポーツ
DONJARAS
涙の優勝

ハイパースポーツの優勝記事

## 君は，胴上げを見たか◆◆◆◆◆◆◆◆◆

ふふふ。というわけで，本物のペナントレースでは中日が優勝できるかどうか，非常に微妙なところだが，生中継ではさっさと残り8試合というところで優勝を決めてしまった。写真を見よ。これが胴上げだ。へへへ。

さて，ペナントレースも終了した生中継68だが，いろいろとバグも多い。いまのところ（最新出荷版ではどうかわからないが），打率の計算や防御率の計算に問題があったりする。

それだけならまだいいが，なんと，同率首位でシーズンを終えると（私がZOOMで遊んでいる間にA氏が頑張ったおかげで，バッファローズとライオンズが同率首位で終えたのだ），なんと，プレーオフが行われることもなく，順位表の上のほうにあるライオンズが優勝してしまったのだ。これは問題である。

A氏の苦労は報われなかったのだ。残念でした。

プロ野球のニュースでもおめでとう

そしてはじまる日本シリーズ

### サウンドのよさがやる気をそそる

野球ゲームの音楽というと「応援歌」がメインで，それ以外の曲はどうでもいいという感じがあった。誰も野球ゲームにBGMのよさは求めないし，たとえ悪くても文句はいわなかった。

ところが，「生中継68」はどうだ？　まず，オープニングデモの曲でア然とするだろう。なぜ，こんなにかっこいい曲が野球ゲームに？　ゲームセットに流れるテーマはもろT-SQUAREのノリだし，そのほかボコーダーボイスを使った曲もあったりして，妙に力が入っている。応援歌だって手拍子やら太鼓やら笛やらが鳴っていて，かなり賑やかでリアルだ。いままでのゲームチックなものとは一線を画している。「ファールボールにご注意ください」や打者紹介の場内アナウンスが流れるのもいままでなかった試みだ。

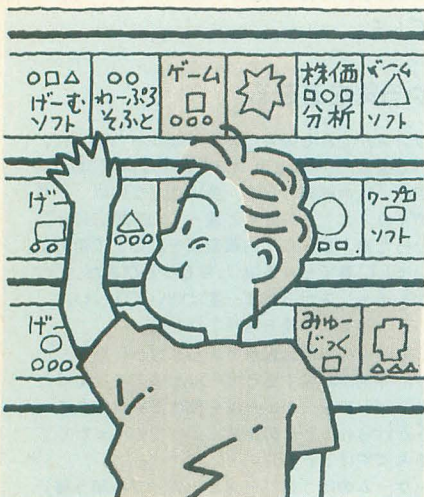
効果音，BGMなどサウンド面ではほぼ100点満点の「生中継68」だが，多少残念なのは「パロディウスだ！」のときと違ってMIDI対応じゃなかった点。しかし，内蔵音源をここまで苛め抜いて(？)奏でられるGMたちもひさびさだ。とにかくチャンスがあれば一度は聴いてほしい。

ところでこの場を借りてひと言，

「勝率や打率の計算がときどきおかしくなったり，守備が暴投するとボールが消えてしまったり，ピッチャーがボールを投げたあと，タイムをかけられるとそのまま止まってしまったりするんですけど」

（ゲームのほうはバージョンアップを願う善）

# AFTER REVIEW

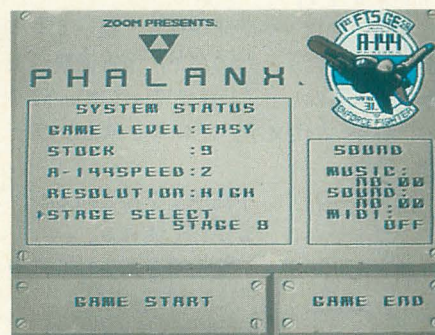
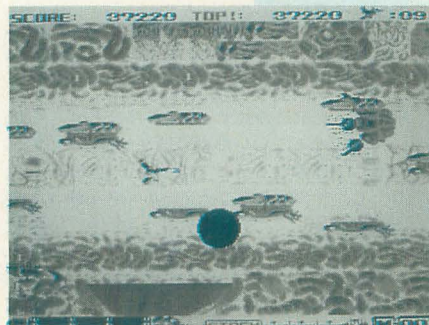
あのズームがシューティングゲーム，ということで注目を浴びた「ファランクス」が今月のお題。ゲームの中身はもちろん，レビューに対する賛否の意見もあって，いつもより多くのハガキが寄せられてきました。

## ファランクス

▶BGMが面白くないとレビューで書いていましたが，私はあれでいいと思います。たしかに面白くない曲もありますが，私が評価しているところは，音色が変わっているということです。たいていのソフト会社は音色を変えていないと思いますが，BGMは大事だと思います。実際，ファランクスの音色が前2作と同じだったら，私はズームファンをやめていたかもしれません。結局私がいいたいことは，BGMはゲームには欠かせないものでしょうから，プレイヤーに飽きさせない曲作りをしてほしいということです。あと不満を少々。西川さんの指摘どおり，3，5，7面はやはりマズイと思います。登場人物もマニュアルには詳しく載っていたのに，ほとんど意味がなかった。コンティニューも制限あり，より無制限だったらと思いました（いちおう，エンディングまでは行った）。

宮本　健司(16)大阪府

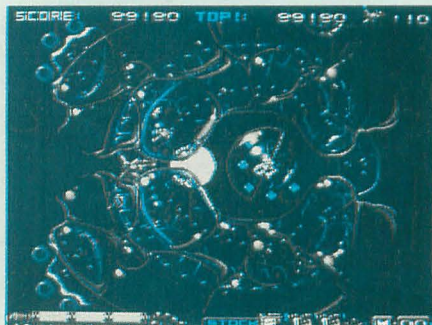
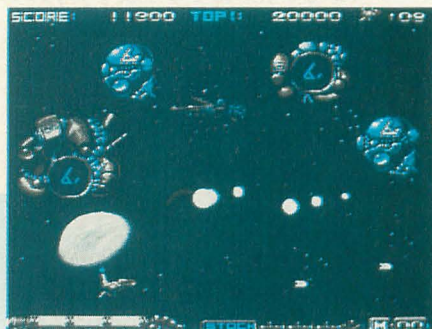
▶レビューでも書いてあったが，僕もファランクスのBGMは面白くないと思う。なんかズームのゲーム音楽はジェノサイド→ラグーン→ファランクスとだんだん悪くなってきているような気がする。ゲーム自体も僕にとってはEASYでもやたら難しいし，なんか全体的にR-TYPEを意識しているような気がして，少し期待外れだった。

渡辺　靖仁(18)三重県

▶シューティングゲームの理想を考えると，やはり射撃というものに快感が直結しているべきではないでしょうか。射った数だけ敵が破壊されるのは非常に当たり前だけれども重要なことだと思います。ところが，このファランクスにはルートを選択させたり，迷路構成のステージといった快感に直結していないものが，ゲームの目的として

存在しているではありませんか。これは考えてみれば非常に興をそがれるものがあると思うのです。やはりシューティングはシンプルが一番。ズームなら純粋なやつでもすごいのができるはずですから。

山口　義英(24)東京都

▶5月18日に買いました。オープニングディスクを見て，ズームの技術力の高さを身をもって知りました。そして，いざゲームを始めると，EASYとNORMALモードでは結構遊べたのですが，HARDでは99回のコンティニューがあるにもかかわらず，1面もクリアできませんでした。なんと情けない。トホホ……。　平谷　淳一(24)兵庫県

▶「パロディウスだ！」よりも「ファランクス」のほうが僕は好きです。やはり，シューティングはシリアスでなくては，BGMは特に1面のものが渋めで好きです。最近のシューティングにはめずらしく，シリア

スで使命感みたいなものを感じさせるところがいい。でも，MIDI音源を前提に作ってあるのか，内蔵音源では少し弱い気もする。ボリュームは大きいけれど。

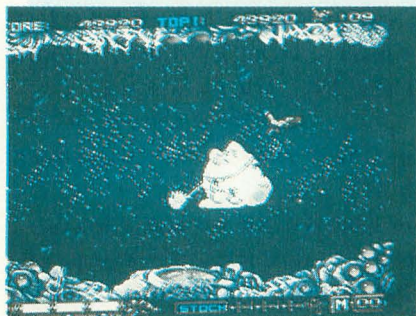
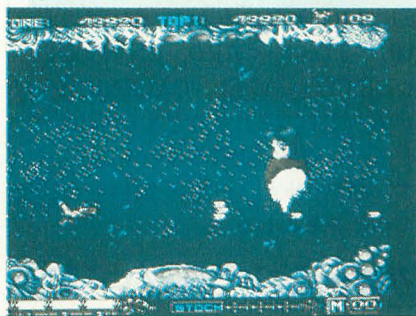
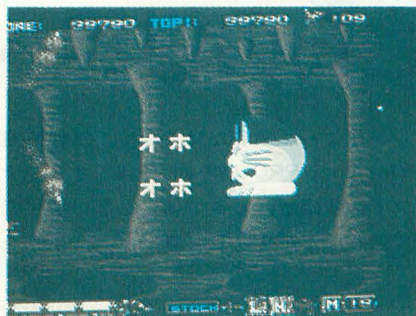
小越　剛志(23)東京都

▶「ファランクス」のレビューに対していいたいことがある。さすがズームというか，グラフィック関係（特殊処理も含めて）は目をみはるものがある。サウンドが悪いという意見には私も賛成である。いままでのズームのゲームに比べると数段劣っていると思う（MIDIでは聞いてないが）。ワープ面に関する意見も同感だが，オープニングなどでの「アニメ顔」はいらないという意見には大反対である。隠し面のネコモドキのほうがよっぽど世界観をこわしているというものだ。

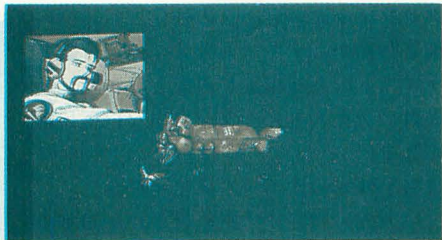
松森　弘樹(19)群馬県

▶おおお，すごいぞ「ファランクス」。なんてったって，あの自機！　超細かい。まさに神業。しかも，動く動く。ドッターの鑑です，あなたは。プログラムだってすごいぞ。技術があるのは当然。センスのよさがやっぱりズーム。音楽だって十分水準以上だ。そりゃ，「ジェノサイド」のほうが好きだけど，これだっていいぞ。3，5，7面がファミコンのゲームみたいでやだってのもあるけど，ほかがよすぎるから許す。ただ，ゲームバランスには少し問題があると思う。ほかのゲームもそうだけど，もっと難易度の幅を広げてほしいぞ。

松本　浩幸(19)大阪府

▶巨大戦艦はうまく使えば面に変化がついていいけど，あれだったらボスを3つ続けて出したほうがまだまし(いいすぎ？)。5面が始まって，5分間何も敵が出ないと思って待っていたのは僕だけにしても，誘導のメッセージとか，敵の一部が見えていても，ばちは当たらないと思うのだが。あと1面の後半と4面の紙芝居のような多重スクロールは個人的にはないほうがましだと思う。悪いことばかり書いたけど，いいことを書くとハガキ100枚でも足りなくなるので書かないだけで，プレイしたことがある人ならそんなことは書くまでもないことはわかっているでしょう。特に6面の「GENOCIDE IV」とボスの触手の下に描いてある絵と，8面のモドキは笑った

大山　和紀(17)静岡県

▶パワーアップウェポンのハデさはまさにズーム。内容もバランスが取れていてよい(ズームにしては簡単か？)。でも，圧搾弾が強すぎでは？　ボスが一撃で沈むぞ。さらに隠し（隠れていない？）も入って，ズームも余裕が出てきたかなと感じさせる。「もどき」のおかげで最終面は一発で終わった。

佐藤　貴是(20)神奈川県

## 発売中のソフト

★オルテウスII　ブラザー工業(TAKERU)
X68000用　5″2HD版 4,800円(税込)
★アクアレス　エクザクト
X68000用　5″2HD版2枚組8,700円(税別)
★ロードス島戦記　ハミングバードソフト
X68000用　5″2HD版3枚組 9,800円(税別)
★銀河英雄伝説IIDX＋kit　ボーステック
X68000用　5″2HD版 5,000円(税別)

## 新作情報

★ボナンザブラザーズ　シャープ
X68000用　5″2HD版　9,000円(税別)
★機動戦士ガンダム クラシック・オペレーション　ブラザー工業(TAKERU)
X68000用　5″2HD版 7,100円(税込)
★キャメルトライ　電波新聞社
X68000用　5″2HD版 価格未定
★スターウォーズ　ビクター音楽産業
X68000用　5″2HD版 7,200円(税別)
★満開電飾　ビクター音楽産業
X68000用　5″2HD版 7,800円(税別)
★ノア　M.N.M. ソフトウェア
X68000用　5″2HD版 7,200円(税別)
★フューチャーウォーズ　スタークラフト
X68000用　5″2HD版 9,800円(税別)
★マジックキャンドル　スタークラフト
X68000用　5″2HD版 9,800円(税別)
★ダーウィンズジレンマ　スタークラフト
X68000用　5″2HD版 9,800円(税別)
★麻雀マスター　ブラザー工業(TAKERU)
X68000用　5″2HD版 7,800円(税別)
★飛翔鮫　金子製作所
X68000用　5″2HD版 予価　8,800円(税別)
★インペリアルフォース　システムソフト
X68000用　5″2HD版 価格未定
★大戦略III'90　システムソフト
X68000用　5″2HD版2枚組 9,800円(税別)
★シュヴァルツシルトII　工画堂スタジオ
X68000用　5″2HD版2枚組 9,800円(税別)
★F15ストライクイーグルII　マイクロプローズジャパン
X68000用　5″2HD版 価格未定
★フェアリーランドストーリー　SPS
X68000用　5″2HD版 価格未定
★スーパー上海ドラゴンズアイ　ブラザー工業(TAKERU)
X68000用　5″2HD版 7,800円(税込)
★SPINDIZZY II　アルシスソフトウェア
X68000用　5″2HD版 価格未定
★ドラッケン　エピック・ソニー
X68000用　5″2HD版 9,700円(税別)
★ゼノン2　エピック・ソニー
X68000用　5″2HD版 価格未定
★シムアース　イマジニア
X68000用　5″2HD版 12,800円(税別)
★パワーモンガー　イマジニア
X68000用　5″2HD版 12,800円(税別)

# ようこそ印刷屋さんの世界へ

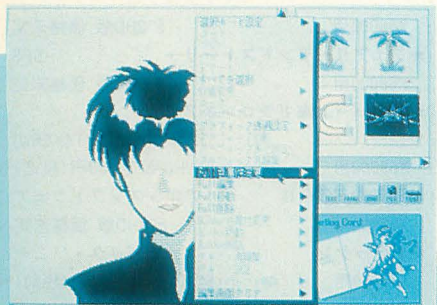
Hamazaki Masaya
浜崎 正哉

バージョンアップによって機能だけでなく，操作感覚も充実したNEW Print Shop PRO-68K ver. 2.0。オリジナル印刷物を作るソフトの決定版，とまではいかないが手軽に誰にでも楽しんで使うことができるソフトだ。

誰でも手軽にオリジナルの印刷物が作成できる「NEW Print Shop PRO-68K」がバージョンアップされました。と9月号でも聞いたようなセリフとともにレビューを始めることにしましょう。

まず，動作環境はメインメモリ2Mバイト必要です。そして，できることならハードディスクがほしいですね。ハードディスクがあれば空き領域をデータのスワッピング領域として使用するため，かなりのデータ処理が可能となります。

さて，目的がオリジナルの印刷物を作成するツールなので，グラフィックセンス，コピーセンスを要求されるのではないか？と心配してしまう人がいるかもしれませんがその必要はありません。もちろん自分で何から何まで作ってしまおうとすると，手間はかなりかかるし，センスもある程度は必要かもしれませんがほとんどの場合，付属しているサンプルデータを使い回すだけで用は足ります。

実際に試してみると操作に戸惑いながらも，30分後にはオリジナルのポストカードが1枚できてしまいました。操作に戸惑ったのは，ろくにマニュアルも見ず，いきなりソフトをいじり始めたからであって，プリントショップが変な操作を必要としていると勘違いしないでくださいね。

①ポップアップメニューで選択

X68000用 5"2HD版4枚組 20,000円(税別)
シャープ ☎03(3260)1161

作成できる項目は全部で，SIGN，BANNER，LETTERHEAD，GREETING CARD，POST CARD，ENVELOPE，CASSETTE LABEL，CALENDARの8つがあります。それぞれどんなものができあがるかは後半で説明していくことにして，まず，プリントショップの全体的な説明をしていくことにします。

## これがバージョンアップしたところ

さて，バージョンアップしたということは，いままでの機能が拡張，変更された(当たり前だ)ということですから，どのあたりが変更されたか見ていきましょう。

1)　グラフィックデータについて
・ver.1.0のグラフィックパーツは直接使用できる。
・ver.1.0の各項目データは直接読み込むことができないので付属のファイルコンバータを使用してver.2.0形式のデータに変更する必要がある。

2)　音声ファイルについて
・ver.1.0の音声ファイルは使用できない。

3)　操作方法について
・マウスのダブルクリック操作をなくし，ポップアップメニュー方式を採用した（写真1)。

4)　機能面の変更
これについては，細かい変更点がありすぎるので一部除いて書いていきます。
・カレンダー，カセットレーベル作成機能が追加された。
・編集画面でグラフィックデータを自由に回転，拡大，縮小できる。
・グラフィックエディタの強化。
・編集画面からテキストエディタを分離してテキストの高速編集が可能となった。
・複数のテキストデータを自由な位置に配置できる。
・文字を任意のサイズで設定可能。
・文字方向の設定に「斜め書き」が追加された。
・編集中にデータを破棄することなくグラフィックエディタ，各項目間を自由に移動できるようになった。
・カットバッファを使って各項目間でデータのカット&ペーストができる。
・複数のパーツを同時に選択して移動させられるようになった。
・レイアウト画面上でパーツを動かすことができる。

とまあ，一部除いてもこれだけの機能変更があります。ひと通り見ただけでもずいぶんと機能が充実しているのがわかるでしょう。もしも，ver.1.0を持っていて活用している人なら，迷わずバージョンアップすることをお勧めします。

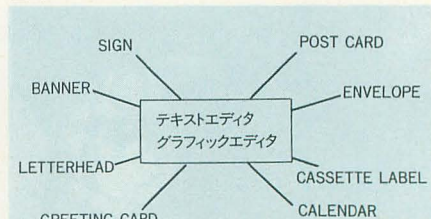
## まずは基本機能から

大雑把にどんな機能があるか見ていくと，各項目は大きく分けて，編集画面，テキストエディタ，グラフィックエディタの3つの機能があり，図1のような関係になっています。中心となるのは編集画面でテキストエディタ，グラフィックエディタを各項目の編集画面から呼び出して使用することになります。

次はグラフィックとテキストの優先順位と，それらがどのように構成されているか説明しましょう。グラフィックパーツは大きく分けて，

1)　Full，Letterhead Full
背景用の固定グラフィックデータ（移動できない)。

2)　Small，Medium，Large

図1

自由に拡大縮小，移動のできるグラフィックデータ。
3) Border
上下左右反転して画面の四隅に張り付けるグラフィックデータ。
の3種類があり，優先順位はFullがいちばん低くBorderがいちばん高くなっていて，テキストは2)と3)の間に配置されます。

で，編集画面に配置されたグラフィックやテキストを移動させるためには，選択状態にしたパーツをマウスでずりずりとひっぱっていくだけです。

移動させるパーツの選択方法は2通りあって，ひとつは個々のパーツを左クリックすると図2のようにベースラインが表示されて選択状態になります。クリックする位置はそれぞれのパーツの中心部分です。ちょっと反応が鈍い場合がありますのでベースラインが表示されるまでしつこくクリックしてあげましょう。

もうひとつは右クリックをドラッグすると破線が表示され，そのままパーツを破線で囲んでパーツを選択状態にするという方法があります。前者は単体の移動，後者は全体の位置調整に使います。

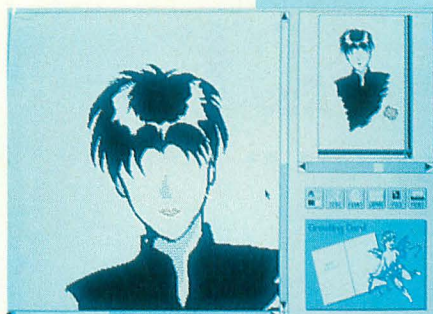
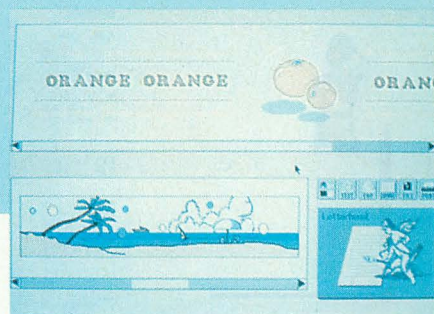
そして，グラフィックの場合はベースラインの角にある丸い点をドラッグしていくとパーツの拡大縮小ができ，テキストの場合は斜め書きの指定がしてあれば，クリックした反対側を中心にして文字列の斜め指定ができます。

## 基本は編集画面

今度は基本となる編集画面はどうなっているのかを説明していきましょう。作業の中心は，この編集画面で行うことになります。

編集画面は写真2のレイアウトとなっていて，編集画面，グラフィックパーツ表示部，アイコン，天使のタイトルが表示されます。ちなみに，SIGN，GREETING CARD，POST CARD，ENVELOPE，CASSETTE LABEL，CALENDARは写真2のようなレイアウトで，BANNER，LETTERHEADが写真3のようなレイアウトです。

ここで重要なのは，右クリックで出現するポップアップメニューで使用できる機能がクリックする位置によって違うことです。ひと目で機能がわかるものはおいといて，以下ずらずらと書き並べていきましょう。

1) 編集画面
グラフィックパーツ，テキストを配置するエリアです。右クリックのポップアップメニューは，パーツのカット，コピー，ペースト，グラフィックの諸設定，Full，Borderの諸設定，テキスト再編集と設定などが行えます。

②SIGNなどの編集画面

2) グラフィックセレクト画面
編集画面に配置するグラフィックパーツを表示するエリアです。右クリックによるポップアップメニューで，セレクト画面に表示するグラフィックパーツの切り替え，グラフィックエディタでパーツの編集，グラフィックパーツの新規作成，グラフィックデータがあるディレクトリの指定，セレクト画面の表示形式の変更が行えます。

3) アイコン
モノクロ/カラー編集の切り替え，テキストエディタへの移動，表面/裏面の切り替え，全体のレイアウト画面の表示，編集画面のセーブ/ロード，印刷の機能が使えます。

4) 天使のタイトル
どの項目で編集を行っているか表示しているタイトルですが，ここで右クリックして出現するポップアップメニューで各項目の編集画面の移動と全体の終了，編集方向の設定，全角文字の書体設定，サウンド設定を行うことができます。

各項目の編集画面で細かな違いがありますが，操作は統一されていますのでひとつ理解しておけば迷うことはないでしょう。

## 文字装飾はテキストエディタで

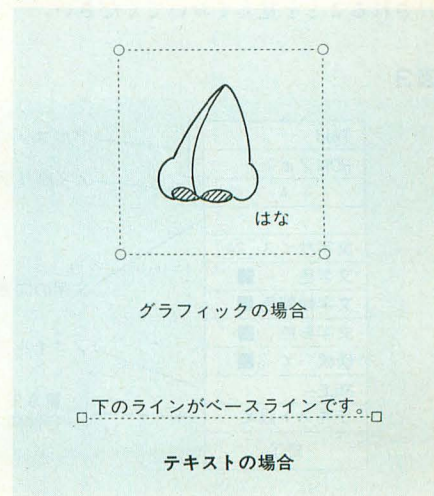
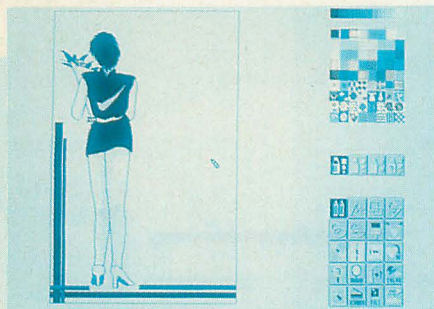
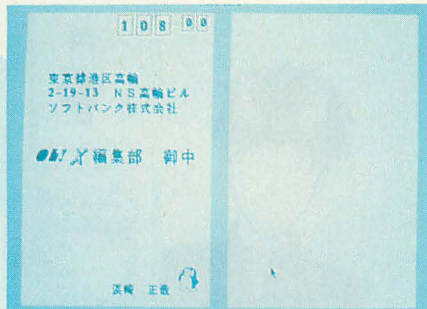
グラフィックの配置は各項目の編集画面で行いますが，テキストについては今回のバージョンアップで分離されたテキストエディタを使用することになります。注意してほしい点があって，TEXTアイコンをクリックしてテキストエディタを呼び出した場合には，新規文章の編集，編集画面のポップアップメニューのテキスト再編集を選択すると，現在選択しているテキストの再編集を行うということです。

こうして呼び出したテキストエディタは写真4のような画面になります。左側がテキスト編集エリア，右側に装飾設定用のメニューが並んでいるのがわかるでしょう。それぞれどのような機能があるかは図3に書いておきました。ちなみにメニューは，現在のカーソル位置のすぐ右側の文字に設定されている装飾設定が表示されています。装飾は文字単位で行うことができるのがポイント。装飾指定したい文字をマウスカーソルでドラッグし，反転させて文字選択をしてから設定を行います。

③BANNERの編集画面

④テキストエディタ

こうして装飾を施したあと，必要ならばテキスト設定で書き方，文字間隔，行間隔，文字方向を指定して終了します。設定した文字は，終了して編集画面に固定してからでないと変化がありません。不安になるかもしれませんが，手順を間違わなければ編集画面に戻ったとき思ったとおりの文字が表示されることでしょう。もしも，指定を

図2

⑤データを加工するためのグラフィックエディタ

⑥まず，宛先を書く

間違ってしまったらテキスト再編集を選択してもう一度やり直してください。

文字フォントは，標準で半角文字フォントを11パターン，全角文字はROMフォントのみサポートしています。また，Zeitの書体倶楽部に対応していますので，アウトラインフォントによる美しい文字印刷が可能となっています。

## グラフィックエディタだ

グラフィックエディタにどんな描画機能がついているかは，触ってみればわかるので説明を省かせてもらいます。

基本的にはアイコンを左クリックすることによりその機能の選択，右クリックで選択した機能の諸設定を行います。で，グラフィックセレクト画面のポップアップメニューなどからグラフィックエディタに移動すると写真5のような画面になります。

テキストエディタのように右側が描画アイコン，カラーパレットが並んでいて，左側が描画エリアになっています。そして，描画エリアには，なにやらボックスが描かれています。

別にわざとらしく僕が描いたわけではなく，エディットしようとしているグラフィックのサイズを表しているのです。データの保存をした場合，このボックスの範囲だけされることを覚えておいてください。

## さあ，作ってみよう

ここまでの説明を読んでもらえれば，プリントショップの全体像をつかむことができたと思います。今度はポストカードの制作を通して，実際の作業手順を説明していきましょう。

ポストカードは表面が宛先，裏面が通信欄になっています。とりあえず，ソフトバンク株式会社Oh!X編集部に近況報告をするためのカードを作るとして，最初にやることは宛先の入力です。郵便番号枠を左クリックして郵便番号を入力し，テキストエディタを使い，住所を入力して表面に配置します(写真6)。もちろん送り元の自分の名前も書いておきましょう。それだけではちょっとさみしいと思いますので，Smallのグラフィックパーツをひとつ配置しておきました。

表面が終わったら，今度はアイコンでFRONTとBACKの切り替えを行い裏面の通信欄を書いていきます。はじめにグラフィックセレクト画面をFullに切り替えて，背景にどのグラフィックを配置するか考えます。ここでは，SCREEN5のグラフィックを使うことにしますが，季節は秋。ちょっとこのままでは涼しすぎるので，グラフィックエディタを使ってそれらしいものに加工することにしましょう。また，このグラフィックは全面に描かれていて，文章を配置するとちょっと文字が見づらくなりそうなので，Full再編集からグラフィックエディタを呼び出して文章を配置するところだけ白抜きにしたほうがいいですね。こうして再編集したグラフィックを別のファイル名でセーブし，裏面に配置したものが写真7です。

タイトルはちょっと文字サイズを大きめにして強調なんかで装飾を施し，本文は飾りをつけずシンプルにします。最後に自分の名前を斜め書きにして，グラフィックをちょいちょいと配置，レイアウト画面で全体の位置調整を行って完成となります(写真8)。

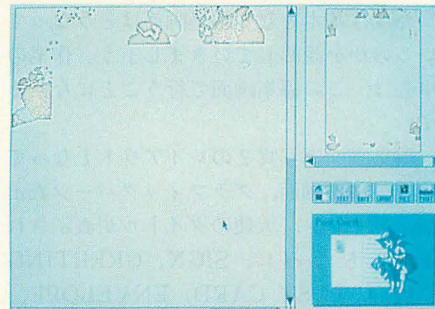
完成したら一応データのセーブをしておきましょう。セーブ項目は，

・DATA……裏面のテキストと表裏面のグラフィックデータ
・ADDRESS……表面のテキスト
・BOTH……DATAとADDRESSの両方

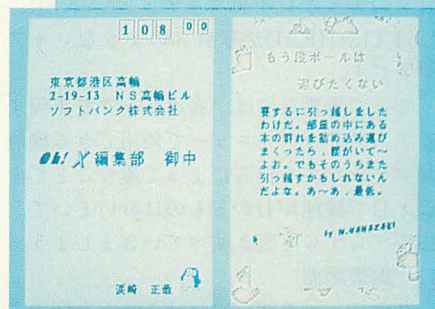
の3つがあります。同じ内容のものを別の宛先で何枚も印刷したい場合には，DATAとADDRESSを別々にセーブしておくと便利だと思います。今回は特定の相手に1枚だけ印刷すればいいのでBOTHを選択してセーブします。

あとは印刷するのみ。これは自分が使用するプリンタ，表/裏面，印字枚数などを選択して印刷を開始するだけです。

## これが印刷例

では，各項目で実際どのようなものが印刷されるのか簡単に紹介しましょう。内容

⑦使用目的にあわせてデータを加工する

⑧全体のバランスを考えて完成

図3

についてはあまりつっこまないように。

・SIGN（ポスター），図4

A4サイズの大きさでポスター（ちらし）の印刷を行います。

・BANNER（横断幕），図5

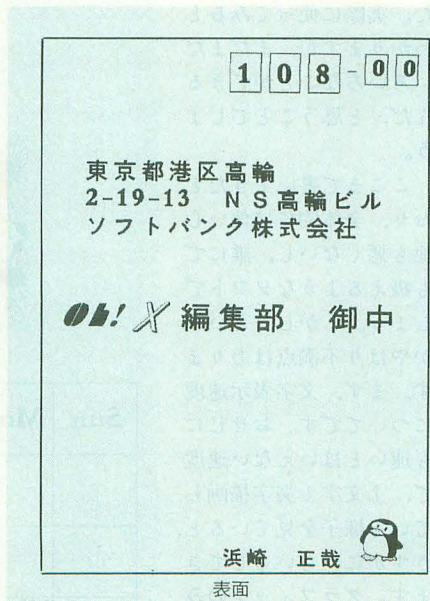
連続用紙を使って横断幕，垂れ幕を作れます。ここでは，目一杯文字を拡大して表示することになるので，ROMフォントをスムージングした文字では，かなり見劣りしてしまいます。なるべくならアウトラインフォントのきれいな文字を使いたいですね。

・LETTERHEAD，図6

見てのとおりB5用紙で便箋を作るものです。エディットするのは便箋のHEADとBOTTOM，つまり上下の飾り部分のところだけです。文章は印刷のときの特殊設定にある，テキスト差し込み印刷モードを使用することで，ファイルにあるテキストを印字することができます。

・GREETING CARD，図7

A4用紙を折り込んで図8のような4面のカードを作ることができます。Fullのグラフィックデータの配置が4通りあること以外は，ほとんどポストカードと同じです。

・POST CARD，図9

先ほど説明したとおり，オリジナルハガキを作ります。

・ENVELOPE（封筒，ディスクエンベロープ），図10

A4用紙で封筒とディスクエンベロープを作れます。写真を見ればわかるとおり，ポストカードの流用といった感じです。違うのは，表裏面が隣接していることと，糊

図4

図6

図7

図8

図9

表面

裏面

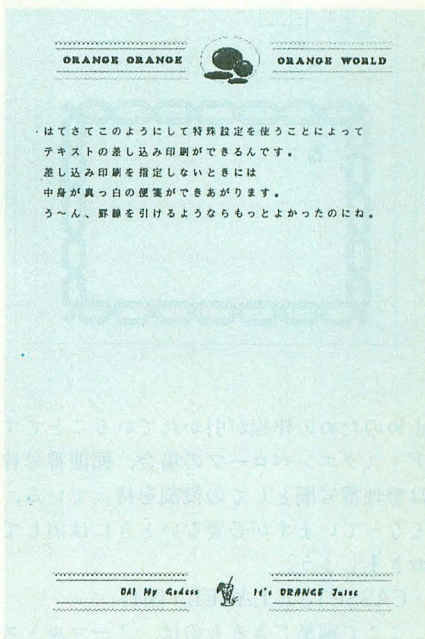
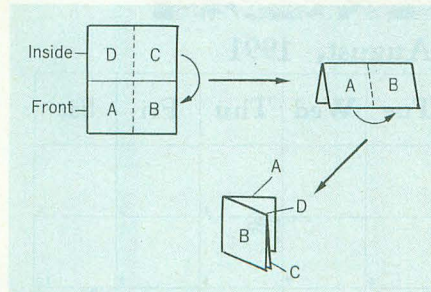



図5

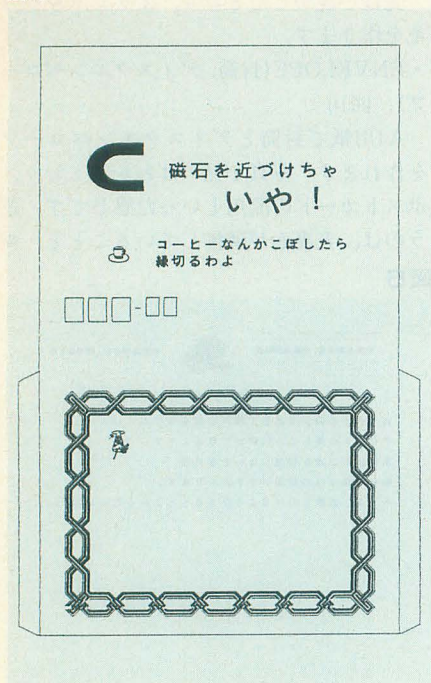
横浜翠嵐高校　電気科学部

「それは我々に対する挑戦だな！」

図10

C 磁石を近づけちゃ
いや！
コーヒーなんかこぼしたら
縁切るわよ

□□□-□□

図11

止めのための枠線が引かれていることです。ディスクエンベロープの場合，郵便番号枠は整理番号欄としての役割を持っている，となっていますが必要ないときには消しておきましょう。

・CASSETTE LABEL，図11

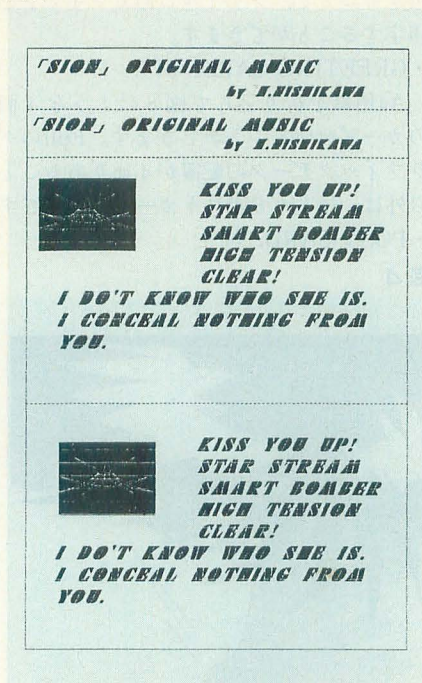
ここで編集できるものは，ノーマル・スリムサイズのカセットレーベル，8mm，VHS-Cのカセットレーベルです。編集画面には折り目の点線が引かれていますから，それに合わせてグラフィック，テキストを配置しましょう。

・CALENDAR，図12

メインはカレンダー設定アイコン。形状を箱型，縦長，横長の3通り，大きさが大，中，小3通り設定でき，縦長の中，大の場合には毎日のスケジュールを記入でき，箱型の大の場合にはさらにSmallのグラフィックまで配置ができます。

ちなみに印刷結果は，すべてVP-800のモノクロ出力で行いました。また，これらの各項目ごとに細かな制限事項があります。詳しいことはマニュアルを参考にしてください(と逃げるヤツ)。

## さて，使い心地は？

ひと通りプリントショップを使ってみた僕の感想は，「面白かった」という言葉で表せます。なにしろこういったソフトを扱うのが初めてなこともあり，触っていくたびに「こういうこともできるんだ」という驚きの連続でした。操作に戸惑っていらいらしたこともありましたが，コツをつかんでしまえば非常に快適に作業を進められます。サンプルデータもディスク2枚にわたって豊富に収録されているため，わざわざ自分でデータを作る必要もなく手軽に作業ができますしね。

また，今回は全体的な説明を中心にしましたので細かいところまで説明しきれませんでした。実際に使ってみるとわかりますが，まだまだいろいろなことができるんだ，と思うことでしょう。

ここまで書いてきたとおり，全体的には使い心地も悪くないし，誰にでも扱えるようなソフトでしょう。しかし，というかやはり不満点はあります。まず，文字表示速度についてです。おせじにも速いとはいえない速度で，1文字1文字描画していく様子を見ていると，さすがにいらいらしてきます。グラフィックのみを扱っているときには快適だった操作が，テキストを配置したとたんに重くなってしまうのです。編集画面での文字表示に，ROMフォントスムージング処理なしの文字表示モードをつけてほしいですね。確かに，画面上ですぐに効果を確認できなくなるというデメリットがありますが，文字装飾は印刷するときに確認できればそれでいいと思いますから。

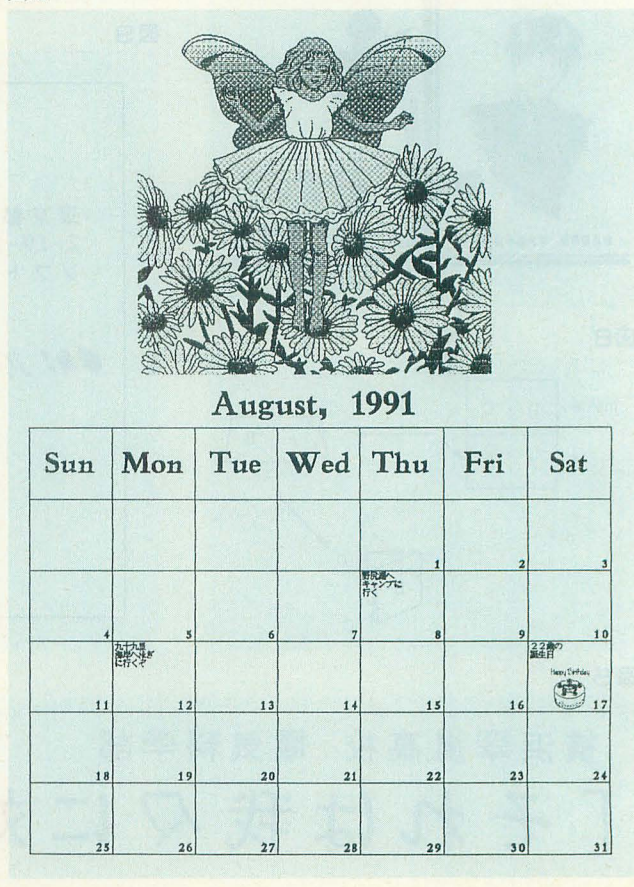
お次はグラフィックエディタについて。付属のグラフィックエディタと考えれば，機能面での充実を見るかぎりよくできていると思えますが，実際にこのエディタで作業をしていくと，使い心地はあまりいいとはいえません。なにか肝心なところが抜けているためこのような感想をいだいてしまうのです。

特に，画面上からのタイルパターンの取り込みとペン描画の重ね描きモードがないのはいけませんね。ディスクに収録されているサンプルデータは，結構きれいなものが多いため，そこで使われているパターンを使いたくてもわざわざ自分でドットの並びをルーペで見ながらパターンを作成する気にはなれません。ちょっとわがままな要望だと思いますけど。

では，ver.2.0になってずいぶん使いやすくなった「NEW Print Shop PRO-68K」。使い方は人それぞれ，皆さんもがんばって楽しいオリジナルカードを作ってみてはいかがですか。

図12

August, 1991

| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|
| | | | | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 野尻湖へキャンプに行く 8 | 9 | 10 |
| 11 | 九十九里海岸へ泳ぎに行くぞ 12 | 13 | 14 | 15 | 16 | 22歳の誕生日 Happy Birthday 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |

# 脳の欲望が指先を動かす

Ogikubo Kei 荻窪 圭

よりよいものを要求するには，よりよいものを知らなければならない。ワープロもしかり。いろいろなワープロを知ってこそ，理想のワープロの姿も見えてくる。というわけで，今回はそのあたりを探ります。

大人の世界では，というより法人の世界では，パソコンは「生産性を上げるための道具」として喧伝され，大勢が真に受けて，導入した。おかげでパソコンは売れ，ここまで普及したわけである。しかし，我々(っていうのは，私や本誌の読者を指しているのだが）は生産性を上げるためにパソコンを購入しているわけではない。では，なぜ我々はパソコンを購入し，あまつさえそのために時間と金を割いてつきあうのか。

西垣通氏の「デジタル・ナルシス」という本（これは非常にユニークな本なので，コンピュータを勉強している人や，チューリングやノイマンに興味のある人は読むといい）に面白いことが書いてあった。

“どうやら人間は「物事を記号化・形式化する烈しい希求」を持っているらしい”というのだ。さらに引用しよう。

“<機械>とは<自然>に対立するものではなく，むしろ人間という生物に付随した「自然の一部」なのである。もしイルカが機械を作るとしたら，それは人間の作るものとはまったく違っているだろう。

繰り返しになるが，機械はユートピアと同様に人間の<形式化への希求>のあらわれである。我々がなぜ<形式>を求めるのかというと，それは<形式>によってコミュニケーションが可能となるからである”

本書には，情報処理世界の先人たちが何に取り憑かれ，どうしてそういう成果を挙げたか（ないしは，どうしてそのようなアプローチをしたか）について書いてある。テーマは，“情報機械とのコミュニケーションスペースに耽溺していく”というデジタル・ナルシスだ。

個人の裁量でパソコンを扱う我々も，大なり小なりそういったものに取り憑かれている。パソコンは便利だとか，パソコンを生活の中に応用しようとか，生産性を上げようとか，パソコンは実用的でなければならないといった大義名分は，“情報機械とのコミュニケーションスペースに耽溺”，あるいは“形式化への希求”という本能的な欲望の“言い訳”にすぎない。でなければ，誰があんな面倒で金のかかる機械を買うものか。ひらたくいえば，パソコンそのものが面白いから買ったのである。

実用的にパソコンを使う人もいるだろうが，我々は「役に立つパソコンを買った」のではなく，「買ったパソコンを役に立たせた」のだ。

しかし，「役に立つパソコンを買う」人もいる。現に，多くのパソコンはそれで売れている。いまのパソコン界は実のところ，結構停滞している。こういう人が増えてきたからだ。なぜなら，彼らに必要なのは，とにもかくにも生産性を上げる，ないしは投資を回収することであり，コンピュータの可能性や深淵にわくわくするようなソフトはどうでもいいからだ。

「大人のためのX68000」は，もちろん，“デジタル・ナルシス”な人々のためにある。パソコンが役に立ちさえすればいいという“成果を求める人々”のためにあるのではない。

またもや長い前書きになってしまったが，要するに，Multiwordなのである。いいソフトには指がキーボードに貼りついたり，マウスが腕の一部になるような感覚を覚えるものだが，Multiwordにはそれがないのだ。Multiwordに限った話ではない。最近の多くのソフトに見受けられることだ。はじめにスペックありき，だったのではないか，という気がする。

我々にとって，アプリケーションに限らず，パソコンとなにかやりとりするすべてがそうだが，“情報機械とのコミュニケーションスペースに耽溺”させるべきものでなければならない。パソコンとのコミュニケーションにおいて，我々は刺激を受け，その作業がひとつの世界を形づくらねばならない。ただ出力を得るためだけに使うのだとしたら，それではコンピュータではなくただの道具であり，その作業は苦行となるだろう。ハイパワーなユーザーは自らの手でそういった環境を構築していく。しかし，そこまでパワーを持ったものたちばかりではない，というのが現状だ。私にだって，そんなパワーはない。だいたい，みんながみんなパワーを身につけて，自分で自分の環境を作るようになったら，それは知恵の無駄というものである。

さて，こういった機械との対話はなにもパソコン相手にだけなされるものではない。車好きは明らかに車と対話しているし，ツーリングを趣味とするものはバイクと対話している。カメラと対話するものもいる。

パソコンの世界であれば，人によって刺激を受けるマシンやソフトは違う。泉大介氏はmicroEMACSと対話しているし（microEMACSを使っているときの彼の指は，傍から見てもキーボードに吸いついているとしか思えない），浦川氏は最近，ストリートファイターIIの筐体と対話している（彼はシャドウストリートファイターIIを行う）。そして，……あまりやると胡散臭くなるからやめよう。

Multiwordはどうか。Multiwordは我々の創造力を刺激してくれるか。“形式化への希求”を満たしてくれるのか。

Multiwordは非常に機能の多いワープロである。DOSの世界では，P1.EXEに迫る機能の多さである。しかし，機能の増加によって得られる出力が向上したとしても，それが創造力を刺激し，滑らかな文書作成を妨げない対話性を持っているかどうかはまったく別の問題である。

今回はそういうわけで，ワープロについてでっちあげてみたい。

## 理想のワープロとは

理想の女と理想のワープロはいつになってもめぐりあうことがない。どちらも技術的には可能なものなのに，である。そして，いつかはクラウン，じゃなかった，いつか理想のワープロが登場したら，いつでもそいつに買い替えるぞ，という決意の下，とりあえず一太郎を使うのである。かくして，みんな一太郎を使い続けるのだ。一度固まった環境は，それがどんな時代遅れになってもなかなか変わらない。革命にさらされないかぎり。まあ，私は一太郎を使っているわけではないので，これは一般論だ。

一太郎の話をするわけではないぞ。理想のワープロの話だ。理想の女を文章にするのは不可能でも，理想のワープロならなんとかなる。

出発点はMultiwordとSoloWriter Ver1.1だ。「SoloWriter Ver1.1」というのは，最近その筋で人気のMacintosh用日本語ワープロだ。私にとっては，いまのところいちばんお気に入りのワープロである。あくまでもお気に入り，というだけであって，理想のワープロではない。いろいろと問題点はある。さらに，Ver1.1でなければならない。Ver1.0は対象外である。詳しくは囲みを参照のこと。

今回SoloWriterをひきあいに出すのは，まったくの偶然である。68,000円も出して買ったかいがあった，という程度だ。が，DOSマシン上に理想のワープロが登場するとは夢にも思えないので，それはそれで妥当な線だろう。

## ワープロとは

いうまでもないことではあるが，いうまでもないことをいうまでもない，とかいって，いわなかったりすると，全体のバランスが変になってしまうので，いうまでもないことでもいう。

### SoloWriterとは

SoloWriterというのは，京都にあるマーキュリーソフトウェアが今年発売した，Macintosh用日本語ワープロである。NISUSという超多機能英文ワープロのVer3をベースに日本語化し，改良を加えたものだ。もとがエディタ出身であるからして，なかなか，その筋のワープロである。あまりに改良したためにNISUSという名前が使えなくなったらしい。SoloWriterにはVer1.0と1.1があり，1.0は日本語処理の部分に問題があり，私の趣味ではない。

SoloWriterの機能は多すぎて紹介に困るほどだが，私の趣味からいって，ポイントは次に挙げる点である。

1）　プルダウンメニューで示されるすべての機能に，ショートカットキーを設定することができる。つまり，コマンドキーを併用したキーカスタマイズなのだが，これが，マウスでちょちょちょいとできてしまうのがすごいのである。なおかつ，どのキーにどのコマンドを割りつけたかのリストも作ってくれる。自分で作ったマクロにキーを割り当てることもできる

2）　行番号をつけることができる。なおかつ，行番号つきの印刷までできる

3）　キーマクロのほか，簡単なマクロのプログラミングもできる。キーマクロといっても，マウスでちょいちょいと選んでやればいい。マクロの名前によっては，オープン時に自動的に実行されるマクロなんてのもできる。作成したマクロは，マウスで普通のコマンドと同様に実行できる

4）　超強力な検索機能を持っている。強力検索モードにすると，マウスで行頭の数字を探せ，なんてことが簡単にできる。さらに，強力検索＋っていうモードにすると，正規表現的な（正規表現かもしれないが，私は正規表現をよく知らないので断定はできないが）メタキャラクタを使うことさえできる。検索は結構速い

5）　ウェブスターの類義語辞典を持っている。当たり前だが，英語のみである。類義語辞典というのは，ある編集部では連載のタイトルをつけたりするのに重宝しているらしい

6）　ルーラーやユーザー定義字体を設定できる

7）　強力なドローイング機能を持っている。ドローイングではあるが，Macintoshであるから，イメージデータも扱える。さらに，図形を文章が避けたりもできるので，便利である。ドローイングのプレーンでも文字を入力できるので，かなり凝った図版やレイアウトも作れる

8）　2段組どころか8段組までできるレイアウトモードは強力で，印刷領域を紙の真ん中に合わせたり，袋とじ印刷したり，などなどいろんな機能を持っている

9）　目次作成，脚注作成の機能がある

10）いろいろと使いにくい点もあるが，将来のバージョンアップに期待している

11）プログラムサイズが800Kバイト以上ある

12）元が英文ワープロであるから，日本語の処理が甘い。さらに，何文字×何行という書式設定ができないから，ちょっと原稿書きには困る。縦書きもできない

13）さすがに，ちょっと遅い

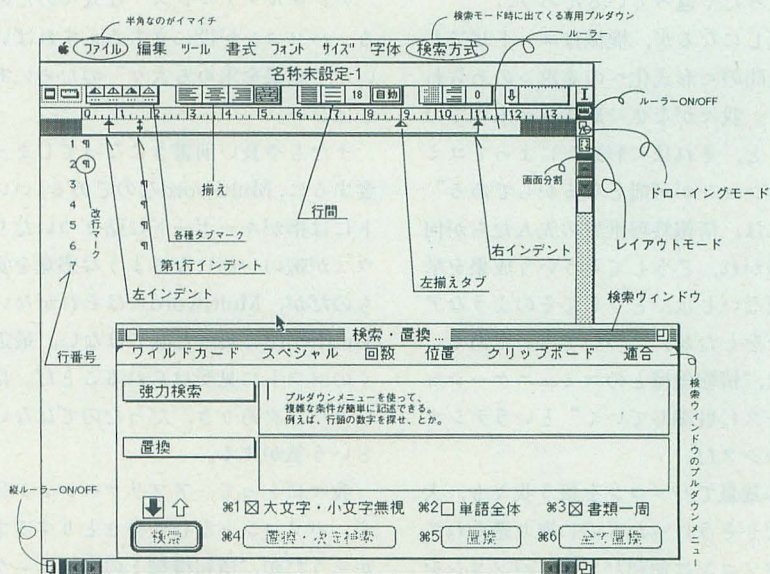
14）最近の舶来のワープロはたいていそうだが，コントロールファンクションをちゃんとサポートしていないから，結構指がホームポジションから離れる

15）まだバグが残っている

16）かな漢字変換処理でけっこうアブないことをやっているために，変換操作に癖がある

17）罫線機能はない（もっとも，ドローイング機能を使えばなんとでもなる)。倍角もない(マルチフォントだからいらない)。網かけもない。結構，英文ワープロでは当たり前だから

と，まあこんな具合であって，そのほかはMacintosh用の普通のワープロと似たようなものだ。これはDTPソフトではなく，ただのワープロである。ワープロとしてはかなり無茶な部類に入るが，実用的な速度は持っており，なかなか私のお気に入りになっているのだ。まあ，時代はここまで進んでいる，と，思ってもらえればいい。DOSのワープロばかりに気をとられているとこういうものを見逃すから。

図

ワープロは，入力・編集・管理・レイアウト・印刷という5つの機能から成っている。それぞれが重要であり，それぞれが問題点を多く抱えている。このなかでどれがいちばん重要かというのは難しい問題だ。個人的にもっとも重要なのが入力・編集であり，続いて管理，となる。この2つがきちんとサポートされていないと，いくら機能が豊富でも使う気にはならない。

ここで視点を変える。1つひとつの機能を追いかけていると，いままでのワープロの延長線を越えることはできない。機能の幕の内弁当になってしまうだけだ。どんなワープロがいいか，と問われて機能を並べ上げることは誰にでもできる。いま，さまざまなアプリケーションが直面しているのは，そういった問題ではないのである。

いかに指をキーボードに吸いつかせるか，が，問題なのである。

## 気持ちいい文書作成

エディタというものがあって，機能はたくさんあって，コンパクトで，速くて，マクロも書ける。しかし，である。毎日何時間も使ったり，マクロ化したいような複雑で機械的な手順を頻繁に使う人はそうは多くない。プログラマならともかく，普通の人はいかに使いでのあるコマンドであっても，週に1回しか使わないものを覚えたりはしたくない。逆に，覚えなくても探せるコマンド体系というのは，Multiwordのようによく使うコマンドが階層の深くにあってうっとうしい。まず両者を揃えるのは必要だろうな。マウスを使ってメニューから機能を選択する体系と，必要な機能を必要なキーに割りつける機能だ。Multiwordは一応キー割りつけの変更は可能だが，コントロールファンクションだけなので，いささか弱いし，いちいちキーカスタマイズテーブルファイルを書き換えるのはインタラクティブでない。

さらに，入力したり編集したりという動作は，日本語FEPを無視しては語れない。

日本語FEPはそれなりに重要な存在だが，これに対する文句は，どれだけ快適な変換をしてくれるかにかかっているだろう。そんなに頭がよい日本語FEPでなくても，変換の操作とレスポンスが快適で，最低限の辞書があれば（ASKの辞書は最低限にも達していない。あらゆる文字が何らかの変換動作で変換できないようでは困る）許す。どうせ，完璧なFEPは望めない。

問題は，そういった動作を支援する機能をワープロが持ちうるか，ということだ。英文ワープロにはスペルチェッカやシソーラス（類義語）辞書を持っているものが多くある。これは立派な入力を支援する機能だ。日本語ワープロでも，SX-WINDOWでお馴染みの文字テーブルや，ちょっとした類義語辞書を用意すべきだろう。富士通のようにCD-ROMを用意しなくても，テキストデータだけならハードディスクで十分だ。何からなにまでうまくいくことはない。じゃあ，うまくいかなかったとき，どう落とし前をつけてくれるか。これが問題なのである。本棚のワープロ用語辞典やマニュアルのコードテーブルを必要としているようでは，何をかいわんや。

ついでに，表示についても触れておこう。私が考えるようなワープロは，どうしてもビットマップになってしまう。そうなると，表示速度が問題となる。これは，多彩なカーソル移動コマンド，簡便なジャンプ機能，高速な検索機能などである程度は補完できるだろう。スクロールは遅くても描きかえが速ければなんとかなるものだ。

挿入や削除の速度が使っていていちばん気になるので，そのあたりをうまく処理してもらえなければ，スクロールだけが速くたって，フォークボールを投げられない野茂のようなものだ。表示速度を補うために，エディタがよく持っている，カーソル位置が画面中央に来るようなコマンドや，microEMACSのようにカーソルが最下行に来たら半ページだけスクロールする，という振舞いも有用だろう。エディタのいいところは速いことではなく，書くことに特化したさまざまな工夫がなされている点だ。microEMACSは図1のように数多くのカーソル移動を支援する機能（検索を含む）を備えている。見習うべきだ。

カーソル移動やカット＆ペースト，各種変換（大文字小文字変換など）の気持ちよさがエディタが好まれる理由のひとつだろう。私は同様な理由で，MacintoshやSX-WINDOWのカット＆ペーストや置き換え入力を気持ちいいと思う。馬鹿だから，このくらい親切でないと間違えるのだ。

続いて，日本語独自の世界に基づいた種々の機能も必要だ。たとえば，エディタの多くがサポートしている単語ごとのカーソル移動は無理にしても，句読点を追って移動する，とか，簡単な文法チェック（句読点や括弧など），半角/全角チェックくらいはしてくれてもいい。ついでに，状況によって追い出し禁則とぶらさがり禁則を選択できるという禁則処理というのもいい。

さらに，画面上で縦書きというのも捨てがたい魅力である。英文ワープロの真似もいいが，いい加減，日本人の文化足りうるワープロというものが必要なところだ。

ある程度の機能を確保したら，欠かせないのがキーカスタマイズやマクロである。これが気持ちよくできないと，濁った東京湾から宝物を探すようなシステムになりかねない。しかも，思い立ったが吉日方式でなくてはならない。ワープロ上からカスタマイズが登録でき，すぐ反映されるのがいい。マクロも，マクロ専用エディタウィンドウが開くくらいの気合いがほしい。

さらに，自分でカスタマイズしたキーを5分で忘れてしまう私のために，ヘルプ機能の1ページ目に，その時点でのカスタマ

図1

```
Global Bindings:

backward-character          previous-line
beginning-of-file           previous-page
beginning-of-line           previous-paragraph
end-of-file                 previous-window
end-of-line                 previous-word
end-of-word                 reverse-incremental-search
forward-character           scroll-next-down
goto-line                   scroll-next-up
goto-mark                   search-forward
goto-matching-fence         search-reverse
hunt-backward               set-mark
hunt-forward
incremental-search
next-buffer
next-line
next-page
next-paragraph
next-window
next-word
```

イズテーブルを表示してほしい。

## 楽な文書管理

あまたあるパソコン用ワープロで，いちばんなおざりにされているのが文書管理だ。よくできたものでさえ，せいぜいファイルのウィンドウが開く程度である。ワープロが文書を作成するものであるなら，作成や印刷だけでなく，その管理まで世話してくれるのが筋というものであろう。ディレクトリ作成機能や複数ファイルを指定して，任意のディレクトリに移動したりコピーしたりする機能は当たり前。ワイルドカードも使えなければならない。これらは，一連のファイル管理ウィンドウで扱うのがいいだろう。さらに後述するが，文書管理も可能なカタログシートというのもいい。

## 印刷である

WYSIWYGという言葉はすっかりお馴染みである。WYSIWYGモードはあっても罪はない。問題は，視覚的にレイアウトが確認できる状態でレイアウトの設定ができることだ。レイアウト画面に対して文書幅を変更したり，段組を変えたりできないと，意味はないだろう。このレイアウトウィンドウでは印刷を行うこともできる。印刷は，まあ，CZ-8PC5からPostScriptまで，数多くのプリンタに対応していたほうがいい。ただ，カラー対応が必要かどうかは疑問だ。

個人的な要望を伝えるなら，明朝とゴシック（さらに，明朝は細明朝と中明朝，ゴシックは中ゴシックと太ゴシックくらいはほしいが）をアウトラインフォントで持っていること。PostScriptがいいけど，高いからなあ。アドビがいけない。

あ，SX-WINDOWにMacintoshがSystem 7で採用したTrue Typeを搭載するのはどうだろう。Windows3.0はあきらめたみたいだけど。ツァイトの出している書体倶楽部もあるが，あのフォントは直線で構成されているので，あんましきれいではないのだ。

## グラフィック編集モード

つまるところ，Multiwordのグラフィック機能は，イメージデータを扱うものである。ペインティングソフトであって，ドローイングではない。さらに，図形を文字が回り込むという機能もない。文字を回り込ませたいときには，罫線枠をうまく使うなり，その行だけインデントを変更するなりする必要がある。

ワープロに似合うのは，ドローイングのグラフィックである。画面よりはるかに解像度の高いプリンタでの出力が前提となるからだ。

もちろん，ドローイングツールでも，イメージデータを貼りつけることは可能とする。主な用途は，文書にイラストを貼り込むことではなく，図を挿入したり，飾りをつけたりするものである。間違えてはいけない。

さらに，SoloWriterを使っていて，もっと便利な用途を思いついた。原稿執筆時のメモや注釈に使うのである。テキストで文章を書いているとき，ちょっとしたメモをグラフィックモードで矢印をひっぱって，入れておく。そして，必要なら本文に取り込み，そうでないなら，消してしまえばいいのだ。

## アウトラインプロセッシング機能

アウトラインプロセッシングといっても，それほど大仰なものを求めているのではない。

まずは，階層構造の文書を作成可能にし，下位の構造を折り込んだり，展開したりできること。続いて，それぞれの構造を入れ替えたりコピーしたりできること，というのも追加する。

そして，どのクラスの構造も文書シートをぶらさげることができるようにする。目次専用シートのような扱いを可能とするわけだ。

場合によっては，目次専用のシートはファイル管理機能を持っており，任意の行（あるいは単語）に文書ファイルをぶらさげられるようにしてもいい。そうすると，文書管理が非常に楽になる。もっとも，ワープロを抜けた状態でディレクトリ構造をいじられると困るが。こうなると，目次というよりカタログシートだな。

この機能があると何が便利か，というと，本稿のような連載原稿を書くとき，原稿ファイルの管理が格段に楽になるのである。それだけだ。

## 文字はコードではなく図形である

長らく，コンピュータ界では，テキストは文字コードの並びとして扱われてきた。いや，いまでもそうか。

だが，実社会では異なる。紙に印刷された時点で，文字も絵もイラストも皆すべて，インクの染みなのだ。1つひとつが記号として意味（というか役割）を持っているかどうかの違いだけなのである。

実のところ，文字も図形も同じである。同じ人の顔にも描き方によってピンからキリまであるように，文字も，使うフォントやその大きさによって，得られるイメージが格段に違う。

いままではプリンタ出力された文字，というひとくくりで微妙な違いは気がつかない振りをしていたが，そうではないのだ。大きな違いなのだ。画面出力の場合，その解像度の低さから日本語表示のバリエーションがむずかしいという話もあるが，フォントだけが問題なのではない。文字間や行間の問題も大きい。行が詰まっているのと，開いているのとでは，イメージがまったく異なるのである。

エディタ愛好者はこれらを犠牲にして，自分の思考速度やタイプ速度に対応してくれるレスポンスや操作性を求めた。私は，自分の文章のイメージを崩さずに記述できる画面出力や操作性を求める（あまり思考速度が速くないからぜいたくをいえるのだ）。人生いろいろである。

## ユーザーインタフェイス，略すとUI

というわけで，UI（ウイ！）である(笑)。アウトラインプロセッサは日本では受け入れられがたい，と，いわれる。本当にそうだろうか。アウトラインプロセッサの構造が日本人に向いていないのではなく，アウトラインプロセッサのユーザーインタフェイスが日本人に向いていなかったのではないだろうか。

一太郎はVer4でランクなどアウトラインプロセッサ的な機能を導入した。しかし，

そんな機能は誰も使っていない。なぜかっていうと，一太郎のメニュー構造・操作体系では，そういった新しい機能にたどりつけないのである。もう新しい機能をつけ加える余地はない。どこかで一度覚えた操作はいつまでも通用するようにする，操作性の継承が重要だ，といっている某NECの高山支配人とかがいたが，これは一見正しいようでいて，まったく誤りである。よいものであれば継承されるし，よりよいものが登場すればリプレースされていくのが，歴史というものだ。一太郎やLotus1-2-3の操作体系というのは，もはや過去の遺物なのである。次のバージョンへの足かせとなる操作体系をひきずっていては，ソフトウェアの進歩自体を止めてしまう。操作性の継承は既得権保護のために必要なだけだ。

もっとも，長い間，よりよい新しいインタフェイスが出てこなかったのもまた事実だ。うーん。

X68000の場合を考えてみよう。各社それぞれ，自分のところのユーザーインタフェイスがいいと主張したいだろうが，ユーザーはそれでは困る。なにか中心になりそうなものと考えると，SX-WINDOWしかない。そこで，これから出るアプリケーションは，そのアプリケーションがSX-WINDOWに対応している，していないにかかわらず，SX-WINDOWのユーザーインタフェイスに準拠すべきだろう。

プルダウンメニュー用のメニューバーくらいは認めるが，それ以外の点，たとえば，マウスのボタンの使い方や，ウィンドウ操作などは共通化させるのがX68000のためである。

ユーザーインタフェイスのポイントをいくつかあげてみよう。

ひとつは，画面から得られる情報量。余計な操作をしなくても，必要な情報は得られなければならない。

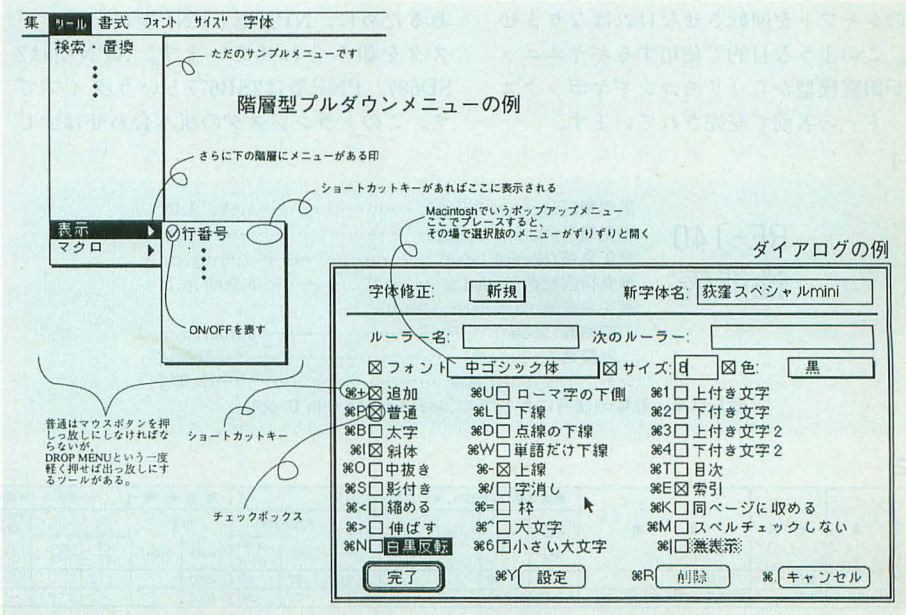
続いて，できるだけ実際に設定が反映する面に対して，直接設定できることだ。たとえば，ルーラーに係わるタブ位置やインデントの設定は，実際にルーラー上でマウスのドラッグによってなされなければならない，ということだ。

さらに，メニューの切り分けである。Multiwordは機能を細かく切り分けすぎた。ほとんどのドロップダウンメニューがさらに子・孫という具合に深い階層を持っているため，頻繁に使う作業をするときも，滅多に行わない作業をするときも同じだけの手間がかかる。これは機能をキーに割り当てられるからいい，という問題ではない。そもそも，マウスを手にした人類に，“文字のカットと行のカット”を別メニューにする意味があるのだろうか。プルダウンメニューやドロップダウンメニューは基本的に1次元である。わかりやすくはあるが，機能が増えれば増えるほどややこしくなっていく。ここで随時ダイアログを用意してしのぐことになる。

メニューに関しては，ワンボタンマウスでここまでやった，というMacintoshの操作体系が参考になるだろう(図2)。これに右ボタンのポップアップメニューが加わったら無敵な気がするのだが，そういう代物はまだ見たことがない。

## Multiwordの抱える問題

Multiwordは速度のほかに，いくつかの大きな問題を抱えている。

ひとつは，多機能にこだわりすぎたことだ。機能を増やすなら，同時に，その機能をいかに使わせるかに腐心しなければならない。深い階層構造の奥に便利なものが隠れていてもしかたがないのである。宝探しが楽しくてワープロを使っているわけではないのだ。

もうひとつは，X68000用のほかのどのソフトともユーザーインタフェイスが異なること，である。

Multiwordを使えば，いままでX68000ではできなかった表現力豊かな文書を作ることができる。しかし，出力を得るためだけに我慢して使うのは苦しい。“情報機械とのコミュニケーションスペースに耽溺”できないのである。

もし，1ページ程度の派手な文書を作りたいだけなら，NEW Print Shop PRO-68K ver2.0を使ったほうがいいかもしれない。

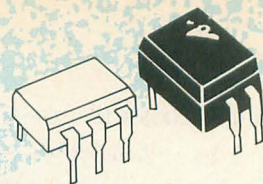
「情報新人類の挑戦」という本に面白い表現を見つけた。これからの時代は，

「“科学的合理主義による人工的世界の無限定的拡大”に対するアンチテーゼの時代」になるそうである。

パソコン界は，“科学的合理主義による人工的世界の無限定的拡大”の終焉を前にして，過渡的な停滞を迎えている。停滞しているからこそ，アメリカ市場がかんばしくないIBMやアップルは，まだ普及の余地がある日本で売れるうちに売っておこうと必死になっているのであり，マイクロソフト社はWindowsという新しいインタフェイスを何がなんでも普及させようとがんばっているのである。

まあ，そんなわけで，いつもとは全然違うノリになってしまった，「大人のためのX68000秋の風スペシャル」は終了する。

図2

ハードウェア工作入門≪16≫

# ハイテクタンク製作（実習編）

Misawa Kazuhiko
三沢 和彦

今月はパトリオットを制御するためのインタフェイス回路を製作していきます。せっかくですから，ただいわれたとおりに作るだけでなく，先月号の理論編を思い出しながら実際，どのように回路が構成されているのか理解を深めていってください。

## 模型用モーターの制御

前回は前後進，左右旋回の可能なハイテクタンク「パトリオット」のモーター駆動部分を設計しました。モーター駆動のための電流回路はステッピングモーターで使用した回路とほとんど同じですし，制御ロジックもTTLIC1個しかないので理解しやすかったのではないかと思います。今月はパトリオットのモーター制御部分を実際に製作するところを説明していきましょう。ステッピングモーターの回路を理解した人には説明が重複しますが，復習のつもりで再度確認してください。

## モーター規格とトランジスタスイッチ

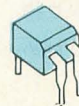
パトリオットで使用したモーターは，マブチモーター製のRE-140というタイプのものです。「パトリオット」は，左右のキャタピラを独立に駆動して左右旋回もできるように設計しました。そのため，モーターを2個並べて取り付け，同じ軸上にある2本のシャフトを回転させなければなりません。このような目的で使用するギヤユニットが田宮模型から「リモコンギヤボックスセット」の名前で発売されています。

ギヤ部分の工作は来月号で詳しく行いますので，今月はエレクトロニクスの部分に絞っていきます。さて，モーターを駆動するにはその規格をチェックしておかなければなりません。表1がモーターの箱に書いてある性能表をそのまま写したものです。ここでチェックすべきなのは，ステッピングモーターの駆動のときにも指摘したようにモーターの消費電流です。RE-140の消費電流は通常使用時で560mAになっているので，この消費電流を十分に流せるようにトランジスタスイッチを設計する必要があるのです。

ギヤとキャタピラを装着すると少し重くなって消費電流が大きくなるかもしれません。しかし，9月号で述べたように今回のインタフェイス回路におけるトランジスタスイッチはステッピングモーターの駆動回路と基本的に同じものです。したがって，650mAが必要なモーターでも十分余力をもって回せたトランジスタ2SD687を今回もそのまま採用することにします。

そして，トランジスタスイッチの部分は正転と逆転の切り替えを可能にしたものであるために，NPN型とPNP型のトランジスタを組み合わせています。NPN型は2SD687，PNP型は2SB677というタイプです。このトランジスタの組み合わせは少し特殊なので説明を加えておきます。

この2個のトランジスタはコンプリメンタリ（相補的）と呼ばれ，2個1組になっているものです。コンプリメンタリというのは，電流の流れる極性が逆になっているNPN型とPNP型に対して，極性が逆である以外は最大定格，電流増幅率など電気的特性がすべて同じもののことです。表2にこれら2つの規格をトランジスタ規格表から抜き出してみました。これを見ると，最大定格，電気的特性すべてのデータ値がまったく同じであることがわかります（極性の違いで符号が違うだけ）。

また，コンプリメンタリという欄があって，たとえば2SD687の所を探すと，確かに2SB677と載っています。実際には，このコンプリメンタリは正負両極の入力がある交流信号を増幅する回路でペアで使われることになります。今回の回路では，モーターを回転させるときには，正転時でも逆転時でも必ずNPN型とPNP型の両方を電流が流れていきますから，この2つのトランジスタの特性を揃えておく目的でコンプリメンタリを使用しました。

## 実際に配線する

部品表は表3を見てください。注意してほしいのは，基板自体をこれまでのICB-87からICB-93S-2に変更した点です。今回の回路はトランジスタだけでも8個あり，加えてTTLICも基板に載せなければならないため，これまでのICB-87では収まりきらなかったからです。しかし，X68000との接続に使う10ピンの汎用ケーブルをつなぐコネクタはいままでどおりのHIF3BA10P-DSを使っています。部品はすべてT-ZONEパ

表1

RE-140 性能表

| 項目 | 値 |
|---|---|
| 限界電圧(Voltage Range) | 1.5V, 3.0V |
| 適正電圧(Normal Voltage) | 1.5V |
| 適正負荷(Normal Load) | 5.8g.cm |
| 無負荷回転数(Speed at no load) | 8,000r.p.m |
| 適正負荷時の(Al normal load) | |
| 回転数(Speed) | 5,700r.p.m |
| 消費電流(Current) | 560mA |
| シャフト径(Shaft dia) | 2.0㎜ |

性能は単1乾電池使用にもとづく(Specification with D-cell)

表2

| 型名 | 社名 | 用途 | 最大定格(Ta=25℃，＊印はTc=25℃) $V_{CBO}$ (V) | $V_{CEO}$ (V) | Ic(DC) (A) | Pc (W) | Pc＊ (W) | 電気的特性 (Ta=25℃)［＊印はtyp値］ ICBO(max) (μA) | VCB (V) | hFE (min) | (max) | VCE (V) | $I_C/I_E$ (A) | Vce(sat)(max) (V) | Vbe(sat)(max) (V) | Ic (A) | $I_B$ (A) | コンプリメンタリ | 外形 | 電極接続 備考 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2SD687 | 東芝 | PSW/PA/PD | 60 | 40 | 3 | | 25 | 20 | 60 | 2000 | | 2 | 1 | 1.5 | 2 | 2 | 0.004 | 2SB677 | TO-220AB形 | BCE.Da/R |
| 2SB677 | 東芝 | PSW/PA/PD | −60 | −40 | −3 | | 25 | −20 | −60 | 2000 | | −2 | −1 | −1.5 | −2 | −2 | −0.004 | 2SD687 | TO-220AB形 | BCE.Da/R |

ーツショップで購入しました。個別部品はいつものときよりも多いのですが，まったく同じ回路を２つ作っているので，配線自体は簡単でしょう。実体配線図（図１）を参照しながら，順を追って工作していくことにします。

最初はICソケットをハンダ付けしましょう。３，６，８，11，14，15番ピンはGNDに直付けなので，ソケットの足を内側に折り曲げます。今回の基板ICB-93S-2はIC用に+5VとGNDのプリントパターンが通っていないので，スズメッキ線を横に渡して，そこに折り曲げたソケットの足をハンダ付けしていきます。次に，トランジスタを取り付けます。トランジスタの足は2SD687，2SB677どちらも同じ順番で，正面から見て左からベース・コレクタ・エミッタの順になっています。これを間違えて取り付けてしまうと付け直すのが大変なので，十分注意してください。取り付けたあとは2SD687と2SB677が互いに向かい合わせ，背中合わせになっているはずです。トランジスタの足も配線する先へ折り曲げてからハンダ付けします。

なお，2SB677のエミッタはモーターの電源に直結ですが，これもスズメッキ線を横に渡して一緒にハンダ付けしてしまいます。また，2SD687のコレクタは折り曲げた足の先にちょうどダイオードのカソード（印のある側）とぶつかるので，ダイオードの足も通してから，一緒にハンダ付けするようにします。モーターの電源は6V，1A（あるいは9V，500mA）程度のACアダプタを使うことを前提にしていますので，いま横に渡したスズメッキ線の電源ラインからACアダプタジャックにつないでおきます。

ダイオード及び抵抗はスペースの都合で立てて取り付ける箇所がありますが，取り付け方は図２の通りです。そして，2SD687と2SB677の間の抵抗は横にして取り付けます。

さて，配線で最も混乱しやすいのがTTLICとトランジスタスイッチとをつなぐジャンパ線です。TTLICから出るジャンパ線は，

1)　汎用ケーブルコネクタへ

セレクトＳ（１番ピン）がIOC4，入力1A（２番ピン）はIOC6へ。入力2A（２番ピン）はIOC7につなぐか，スズメッキ線で基板上を伝わせてもよい。

2)　TTLICの端子どうし

入力1A（２番ピン）から入力3B（10番ピン），入力2A（５番ピン）から入力4B（13番ピン）の２箇所。

3)　トランジスタスイッチへ

・出力1Y（４番ピン）から右チャンネル正転入力端子へ。

・出力2Y（７番ピン）から左チャンネル正転入力端子へ。

・出力3Y（９番ピン）から右チャンネル逆転入力端子へ。

・出力4Y（４番ピン）から左チャンネル逆転入力端子へ。

以上の配線が配線ミスの90%を占めるのではないでしょうか。あとでチェックしてみて，モーターが回らない，片方だけ回る，目的の方向と逆に回る，などの誤動作はほぼここの配線ミスが原因と考えられます。

モーターの配線は前作の旧型タンクと同じ延長ケーブルを使用します。そのために

は８ピンDINプラグ・中継ジャックを使います。DINプラグ・中継ジャックについては，８月号を参照してください。

今回のインタフェイス基板から，左右のモーターそれぞれに２本ずつ計４本のケーブルが出ています。実体配線図ではモータ

**表3　部品表**

| 品名 | 数量 | 価格 |
|---|---|---|
| IC用基板（サンハヤトICB-93S-2） | 1枚 | 260円 |
| 10ピン基板用コネクタ（HIF3BA10P-DS） | 1個 | 100円 |
| 直流モーター（マブチRE-140） | 2個 | @140円 |
| 16ピンICソケット | 1個 | 30円 |
| LS157 | 1個 | 60円 |
| 2SD687 | 4個 | @100円 |
| 2SB677 | 4個 | @140円 |
| 10D1 | 4本 | @ 20円 |
| 1kΩ抵抗 | 8本 | |
| ACアダプタジャック | 1個 | 100円 |
| 8ピンDINプラグ | 1個 | 135円 |
| 8ピンDIN中継ジャック | 1個 | 170円 |
| スズメッキ線 | 少々 | |
| ビニール配線材 | 少々 | |

**図1**

ー1，モーター2と書いてある端子です。モーター1，2ともに極性があります。RE-140には青と赤のリード線が出ているので，向きを揃えてつないでください。

今度はプラグの端子接続図を図3に示します。ここで，4番ピンはX68000のIOA0，2番ピンは+5V，8番ピンはGNDにつないでおくのを忘れないようにしてください。今回のインタフェイス回路ではX68000の入力ポートはまったく使用していませんが，「パトリオット」のマル秘システム導入に不可欠なので，反対側のDINジャックにはRE-140のリード線をつないでおいてください。また，2，4，8番ピンにはいまのうちからビニール被覆線を10～20cm程度に切ってハンダ付けしておきましょう。

## 制御ロジック回路

前回の復習も兼ねて，TTLICのLS157のロジックについて説明しておきましょう。

図2

図3

図4

○セレクト入力をLまたはHすることによりそれぞれデータA，データBを選び出力する
○ストーブをHにすることにより他の入力に無関係に出力をLにする

FUNCTION TABLE

| INPUTS | | OUTPUT |
|---|---|---|
| Select | Strobe G | Y |
| X | H | L |
| L | L | A |
| H | L | B |

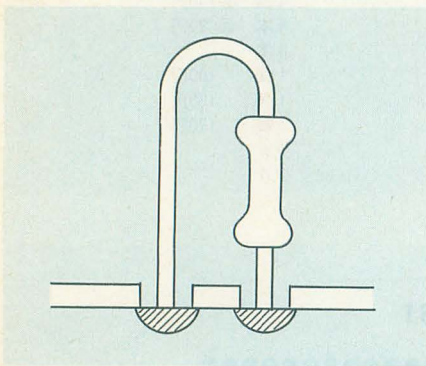
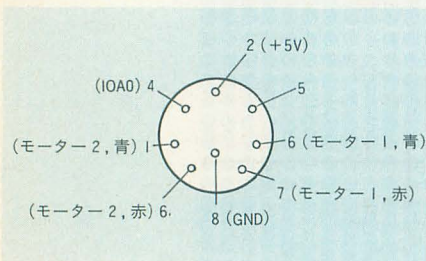
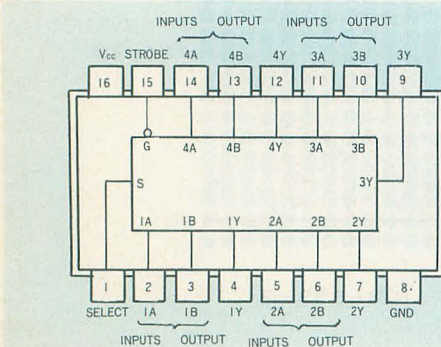



図4が規格表から抜粋したLS157の規格です。このLS157は2→1データセレクタと呼ばれるもので，2系統の入力A，Bに対して，出力Yをそのどちらか一方に選択するスイッチになっています。

データセレクタは入力A，Bと出力Yとを1組とすると，1個のパッケージに4組入っていて，出力の選択はセレクト入力端子（1番ピン）で4組を1度に切り替えます。ちなみにストローブ端子（15番ピン）はパッケージ全体のON/OFFを切り替える入力端子で，ここをLにすると入力A，B及びセレクト端子の内容に関わらず，すべての出力がLになってしまいます。

回路図に即して各端子のロジックを追ってみましょう。IOC4はセレクト入力ですから，LS157の規格表に照らして，LのときY＝A，HのときY＝Bになります。

さて，IOC6は1Aと3B，IOC7は2Aと4Bにつながっています。入力のもう片方は常に1B＝3A＝2B＝4A＝Lとなっています。出力については，CWを正転方向，CCWを逆転方向と表すと，1Y＝RCW，2Y＝LCW，3Y＝RCCW，4Y＝LCCWとなります。以上の約束から表を作ってみると，X68000からの出力IOC4，IOC6，IOC7に対して，下の表のようになります。

| IOC4 | IOC6 | IOC7 | | RCW | RCCW | LCW | LCCW |
|---|---|---|---|---|---|---|---|
| | 1A/3B | 2A/4B | | 1Y | 3Y | 2Y | 4Y |
| （Y＝A） | | | | (1A) | (3A) | (2A) | (4A) |
| 0 | 0 | 0 | ： | L | L | L | L |
| 0 | 0 | 1 | ： | L | L | H | L |
| 0 | 1 | 0 | ： | H | L | L | L |
| 0 | 1 | 1 | ： | H | L | H | L |
| （Y＝B） | | | | (1B) | (3B) | (2B) | (4B) |
| 1 | 0 | 0 | ： | L | L | L | L |
| 1 | 0 | 1 | ： | L | L | L | H |
| 1 | 1 | 0 | ： | L | H | L | L |
| 1 | 1 | 1 | ： | L | H | L | H |

いちばん左のIOC4が，LのときY＝A，HのときY=Bとなります。IOC6＝1A＝3B，IOC7＝2A＝4B，また，1B＝3A＝2B＝4A＝Lとなっていますから，Y＝AのときとY＝Bのときで1～4Yの各出力を写していきます。たとえば，IOC4がLのときは1Y＝1A＝IOC6，2Y＝2A＝IOC7，3Y＝3A＝L，4Y＝4Y＝Lとなります。

出力については，1Y＝RCW，2Y＝LCW，3Y＝RCCW，4Y＝LCCWということから，それぞれのチャンネルに対応する出力を求めることができます。このようにして求めた対応表は，9月号ロジック回路の設計の項で示した表とまったく一致しています（9月号の理論編参照）。

実際の動作についてひとつ注意があります。LS157の出力電流を規格表で調べるとHから流れ出る電流が0.4mAと非常に小さい値になっています。それに対して，C-MOSタイプのHCシリーズは10倍大きい4mAが流せるようになっています。これは，2SD687と2SB677の電流増幅率が最低で2000倍なので，LS157の出力によって0.4×2000＝800mAのモーターまで駆動できることになります。

直流モーターの規格は560mAなのでLS157でも使用可能ですが，少し重いものを駆動しようとすると電流が足りなくなる恐れもあります。そこで，今回のキャタピラタンクで，LS157の代わりにHC157も試してみましたが，動作に変化はありませんでした。気になるようでしたらHCの方を使うことをお勧めします。

## まとめ

うまく工作できましたか？　部品点数がいつもより少し多めなので，実体配線図と回路図とをよくにらんで間違いのないようにチェックしてください。実行させるプログラムはX-BASICのダイレクトモードからIOOUT関数で0から7まで出力させてみるだけでOKです。出力させる値とモーターの回転の仕方との関係は次回までの宿題としておきます。

吾輩はX68000である
[第6回]

# グラフィックモードあれこれ

Izumi Daisuke　泉　大介

吾輩が誇る機能のひとつに
グラフィックの表示能力があげられる
諸兄にも十分に堪能していただきたい

吾輩がほかのパソコンより1歩も2歩も抜きん出ていると自負している機能として，高解像度のグラフィック表示における色数を挙げることができる。最大で512×512ドット×65536色という機能に魅せられてX68000を購入したという諸兄も少なくないことだろう。豊富な色数はゲームプログラムをかつてないほどカラフルで印象深いものにしたし，他方では本格的なパーソナルレイトレーシングを可能にし，多方面のコンピュータグラフィックで活躍するに至っている。

65536色もの色数は必要ないということであれば，1024×1024ドットの大きさを持つグラフィック画面を使用することもできる。実際に画面に表示されるのは768×512ドットではあるが，これにしても640×400ドットよりははるかに大きい。いざとなれば，スクロールさせれば全画面を見ることができるし，使える色数が16色に限定されてはいても，モノクロ16階調(ちょっと悲しいが)なら結構な表現力を期待できる。SX-WINDOWでお馴染みのOh!X式パレットならウハウハである。

残念ながら，うちの御仁はこの方面には全く興味がない風を装っていて，見ることはあっても自分で何かをやろうとするようなことはほとんどない。斜に構えて，「コンピュータをそんなことに使って……」と悟りきったような顔をしている。

実をいえば，これは自分の絵心がないのを人に悟られぬための方便なのである。つい先だっても荻窪氏に触発されて，画像取り込み→再加工というプロセスで「作品」を作る気になったらしく，久しぶりにZ's STAFFを起動したのだが，結局，数時間かけて元の写真を台無しにしただけであった。

絵心とは描写力だけをいうのではない。人を感心させる絵を描くには，構成力も大きな要素を占めている。せめて構成力だけでもあれば，よもやあれほど「芸術的な」グラフィックにはならなかったのではなかろうかと思うのだが。この点に関して荻窪氏は非凡であり，氏の想像力と構成力は名作「都庁バルキリー」となって結実。かくして，御仁は再び方便の殻に閉じこもることになったのであった。

最近では日本で最も幅を利かせている某98シリーズも高解像度への道をようやく歩き始めたようである。ところが，その高解像度の画面に24ドット文字しか表示できないというのである。画面に表示される文字情報量は相変わらず80×25なのだから失笑を禁じえない。

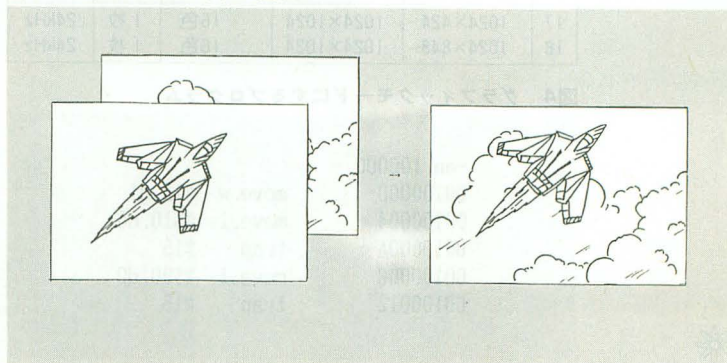
## グラフィック画面のいろいろ

吾輩のグラフィック表示能力は，グラフィック画面のサイズによって以下のように分類できる。

| | | | |
|---|---|---|---|
| 1) | 1024×1024 | 16色表示 | 1ページ |
| 2) | 512×512 | 16色表示 | 4ページ |
| | | 256色表示 | 2ページ |
| | | 65536色表示 | 1ページ |

複数のページを持つものは，それぞれに独立して絵を描くことや描かれているグラフィックを消去することができる。吾輩は複数ページのグラフィックを重ね合わせて表示する能力をもっているので，たとえば512×512ドットの画面，256色表示を選択し，0ページ目に戦闘機を，1ページ目に背景を描くことにすると，1ページ目のみをスクロールさせることによって，あたかも戦闘機が飛んでいるかような効果を出すことができるわけである(図1)。

この1024×1024，あるいは512×512の画面は，図2のよ

図1　2枚のグラフィック画面

うにしてディスプレイに表示される。大きな絵を覗き窓から見ているようなものである。もちろんこれは，表示画面と実画面とで異なるサイズを選択した場合であり，実画面，表示画面ともに512×512ドットを選択した場合には両者はぴったり重なることになる。

表示画面サイズは，256×256,512×512,768×512から選択することができる。グラフィック画面のサイズと表示画面サイズの組み合わせを一覧表にしたものが図3である。

図3は吾輩の自慢である。世の中にパソコン多しといえ

**図2 表示画面と実画面**

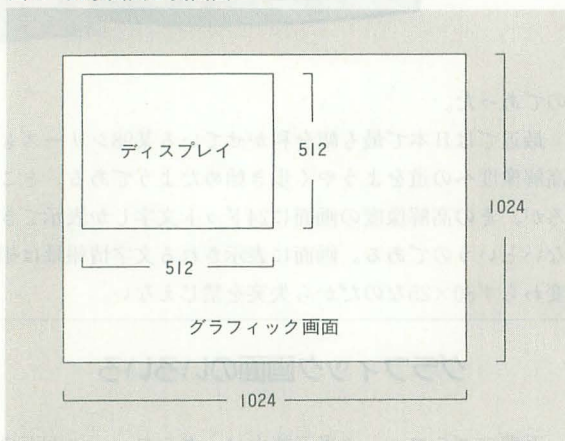



**図3 画面のいろいろ**

| | 表示画面 | 実画面 | 色数 | 枚数 | 周波数 |
|---|---|---|---|---|---|
| 0 | 512×512 | 1024×1024 | 16色 | 1枚 | 31kHz |
| 1 | 512×512 | 1024×1024 | 16色 | 1枚 | 15kHz |
| 2 | 256×256 | 1024×1024 | 16色 | 1枚 | 35kHz |
| 3 | 256×256 | 1024×1024 | 16色 | 1枚 | 15kHz |
| 4 | 512×512 | 512×512 | 16色 | 4枚 | 31kHz |
| 5 | 512×512 | 512×512 | 16色 | 4枚 | 15kHz |
| 6 | 256×256 | 512×512 | 16色 | 4枚 | 31kHz |
| 7 | 256×256 | 512×512 | 16色 | 4枚 | 15kHz |
| 8 | 512×512 | 512×512 | 256色 | 2枚 | 31kHz |
| 9 | 512×512 | 512×512 | 256色 | 2枚 | 15kHz |
| 10 | 256×256 | 512×512 | 256色 | 2枚 | 31kHz |
| 11 | 256×256 | 512×512 | 256色 | 2枚 | 15kHz |
| 12 | 512×512 | 512×512 | 65536色 | 1枚 | 31kHz |
| 13 | 512×512 | 512×512 | 65536色 | 1枚 | 15kHz |
| 14 | 256×256 | 512×512 | 65536色 | 1枚 | 31kHz |
| 15 | 256×256 | 512×512 | 65536色 | 1枚 | 15kHz |
| 16 | 756×512 | 1024×1024 | 16色 | 1枚 | 31kHz |
| 17 | 1024×424 | 1024×1024 | 16色 | 1枚 | 24kHz |
| 18 | 1024×848 | 1024×1024 | 16色 | 1枚 | 24kHz |

**図4 グラフィックモードにするプログラム**

```
-an 100000
00100000        move.w  #12,d1
00100004        move.l  #$10,d0
0010000A        trap    #15
0010000C        move.l  #$90,d0
00100012        trap    #15
```

ども，これほど多彩な表示方法を選択できるパソコンは，そうはあるまい。図3の周波数欄にご注目いただきたい。256×256,512×512ドットの表示画面モードはいずれも15kHzで表示できるようになっている。吾輩が表示するコンピュータ画面をビデオに収めておくのに利用されたい。これらのうち，横方向のドット数が512ドット以下の画面モードでは，スプライトも表示することが可能である。

諸兄のなかには，17,18番目のモードを初めてご覧になる方がいらっしゃるかもしれない。もし，MicroEMACSというエディタをお持ちなら，$sresという変数に17,あるいは18をセットしてみていただきたい。もしかしたら，お手持ちのバージョンは$sresという変数をもっていないかもしれないが，もっているなら画面モードが変更されるのをご確認いただけるはずである。C言語のプログラムを作る際,1行が長くなりすぎて画面がスクロールしてしまう場合には，通常，行を2つに分けるなどの対処を行う。

ところが，呆れたことに，御仁はMicroEMACSの画面モードを，17の1024×424に変更して作業をする。これならば最大で横に126文字も表示できるためである。PASCALがタブを4文字に設定しているのに対し，C言語は8文字を標準としている。これは，行が表示できなくなるほどインデントが深くなるのはアルゴリズムが悪いからだという美学の現れであったはず。御仁の対処は本末転倒もいいところだ。ただ，確かに便利な機能であるには違いない。

## グラフィックを試してみる

合計19もあるグラフィックモードのうちのどれを使用するかを設定するには，IOCSコールの$10_H$を利用するのが簡単でいい。

このIOCSコールは，

```
move.w  #0,d1
move.l  #$10,d0
trap    #15
```

のように使用する。図3の表の最初の欄は適当につけた数字ではなく，選択する画面モードに対応して振ってある。自分の選択したい画面モードを選び，対応する数値をD1.wレジスタにセットして利用されたい。

これだけではまだグラフィックを表示することはできない。どの画面を使用するかを設定したら，IOCSコール$90_H$を使って使用可能状態にする必要がある。こちらはグラフィック画面をクリアし，グラフィックを表示状態にする。IOCSコール$90_H$は，ただD0レジスタにサービス番号をセットして「trap #15」を実行するだけでいい。したがって，全プログラムは図4のようになる。

図4のプログラムは，どうせなので吾輩の65536色モー

ドを試していただこうと，D1.wレジスタに12をセットしている。例によって，諸兄のデバッガのPコマンドが示すアドレスから入力していただきたい。プログラムの入力が終わったら，

　－b0 100014

のようにブレイクポイントを設定し，

　－g=100000

で実行である。くれぐれも注意しておくが，吾輩のグラフィックVRAMをRAMディスクに使用している場合は，このプログラムを実行なさらないように。RAMディスクのデータが消失してしまうことになる。どうしてもRAMディスクを外したくないという諸兄は，必要なファイルをフロッピーディスクやハードディスクに保存してから実行されたい。ただし，デバッガを抜けたあと再びRAMディスクを使うには再フォーマットする必要がある。

**●グラフィック画面に点を打つ**

テキストVRAMにデータを書き込むと画面にドットが表示されたように，グラフィックVRAMにデータを書き込んでも画面にドットが表示される。テキストVRAMでは，2枚あるテキストVRAMのどちらにデータをセットするか(あるいは両方にセットするか)によって色付きのドットを表示したが，ここのところの事情はグラフィックVRAMでは若干異なっている。グラフィックVRAMは図5-1のような構造をしており，1ワード(2バイト)が1ドットに相当する。1ワードなので，0000～$FFFF_H$の65536色が表示可能なのである。

世の中の多くのコンピュータは，グラフィックVRAMに吾輩のテキストVRAMのような構造を採用している。吾輩が白い文字を表示するのにテキストVRAMの2カ所に同じデータを書き込まなければならないのと同じように，彼らはたったひとつのドットをセットするのに何カ所にも(16色モードで4カ所も！)データを書き込まなければならないのである※。

しかも，1バイトのデータが8ドットの横線を表示するようになっているため，1ドットだけセットするには，面倒な作業が必要となる。たった1度データを書き込むだけで65536色を表示できる秀逸のグラフィックVRAMは，吾輩の自慢なのである。

諸兄はさっそくデータを書き込みたくてウズウズしておられることだろう。グラフィックVRAMはアドレス$C00000_H$から始まっている。存分にデータを書き込んでみていただきたい。

書き込むデータの形は図5-2に示したとおりである。データは2進数で考えた場合の形式を示してある。画面左上に緑のドットを表示したければ，メモリにデータをセットするMEコマンドを使って，

　－me c00000

　00C00000　0000：■

としたところで，

　00C00000　0000：_1111100000000001

のようにデータをセットすればいい。「_」はデバッガで2進数を指示するのに使用する。

データを入力するごとに，緑の点が画面左上をノロノロと右へ向かって動き出し，やがて線となるのが確認できただろうか。線の色を赤や青にしても試されたい。もちろん，これらを微妙な割合で混ぜ合わせれば，美しい中間色を表示することができる。

ラインを1つ下へ伸ばしたい場合は，アドレスに$400_H$を加えればいい。横が512ドット，1ドットが1ワード(2バイト)なので，1024バイト＝$400_H$バイトというわけである。この調子でいくとグラフィックVRAMの最後は，$C00000_H$＋512×512×2－1＝$C7FFFF_H$となる。

　－d c7ff00

として$C7FF00_H$以降を表示してみていただきたい。いかがだろうか。$C7FF00_H$はちゃんと表示された？　では続きをDコマンドで表示してみていただきたい。そろそろである。まだOK？　では，その続きを……。いかがだろうか。図6のようになったことと思う。メモリがないため，バスエラーが発生したのである。

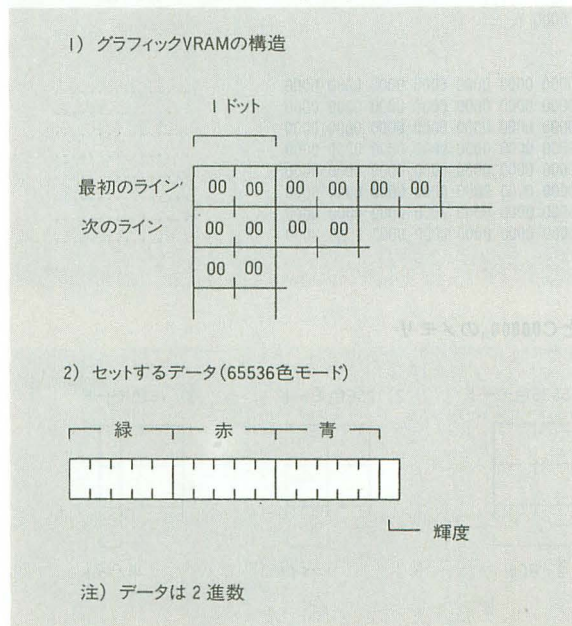
※　X68000では同時アクセスモードやビットマスク機能のおかげでアクセス回数は減っている。PC-9801でもGDCの機能として同様のものがあるが，手順が複雑になるため単純なアクセスではほとんど使用されない。

**●グラフィック画面モードを変えてみる**

グラフィック画面を512×512ドットのモードにすると，文字も1行64文字表示になる。これではデバッガの画面がいささか見づらいと思うので，元の96文字表示に戻すことにしよう。図4の$100000_H$に入れた命令を，

　100000　move.w #16,d1

に変更して再びプログラムを実行すればOKである。同

**図5　グラフィックVRAMとセットするデータ**

時にこれは，グラフィック実画面を1024×1024ドットにするプログラムでもある。さて，ここで諸兄にはもう一度C7FF00$_H$以降のメモリを表示してみていただきたい。先程はC80000$_H$以降にはメモリはなかった。今度はいかがだろうか。

C80000$_H$以降にも，ちゃんとメモリが存在しているのを確認できたことと思う。念のため付け加えておくと，このモードでも1ドットは1ワードである。「おお，X68000は1024×1024×2＝2MバイトのグラフィックVRAMを持っている!!!」という歓声が聞こえてきそうである。そんなわけがあるはずがなかろう。カタログにもグラフィックVRAMは512Kバイトと明記してあるではないか。

試しに，アドレスC00000$_H$にFFFF$_H$をセットしてみていただきたい。セットできたら，今度はDコマンドでC00000$_H$を表示されたい。図7のようになったことと思う。FFFF$_H$をセットしたはずなのに，表示されたデータは000F$_H$となっている。なんと，アドレスC00000$_H$は半バイトのデータしか受け付けないのである。

図6　グラフィックVRAMはどこまで？

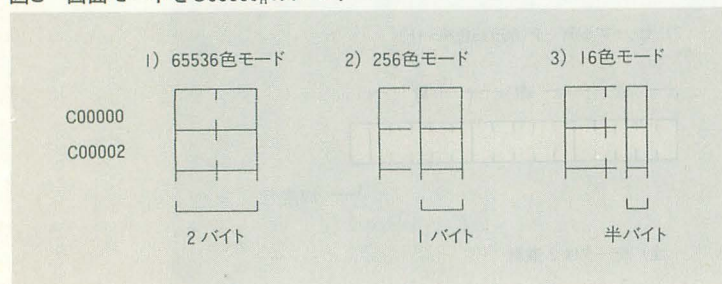
```
-d
00C7FF80  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C7FF90  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C7FFA0  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C7FFB0  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C7FFC0  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C7FFD0  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C7FFE0  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C7FFF0  0000 0000 0000 0000 0000 0000 0000 0000        ................
-d
00C80000

bus error in debugger
```

図7　1024×1024ドットモード

```
-me c00000
00C00000        0000 :ffff
00C00002        0000 :.

-d c00000
00C00000  000F 0000 0000 0000 0000 0000 0000 0000        ................
00C00010  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C00020  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C00030  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C00040  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C00050  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C00060  0000 0000 0000 0000 0000 0000 0000 0000        ................
00C00070  0000 0000 0000 0000 0000 0000 0000 0000        ................
```

図8　画面モードとC00000$_H$のメモリ

この秘密は図8をご覧いただきたい。吾輩の作り主であるシャープ大人は，どんな画面モードでも「1ドット＝1ワード」を貫く，という非常に明確な意志をもって吾輩の創生に当たったのである。だからといって，2MバイトもグラフィックVRAMを搭載するわけにはいかなかった。もしそうなっていたら，吾輩は非常に高価なマシンになってしまっていたことであろう（512×512ドット×42億色を2画面とか1024×1024ドット×65536色というのは，いささか食指が動かないでもないが。これならまさに，総天然ショックである）。

そこで大人は，1ワードのメモリを切り売りするという画期的な妙技を吾輩に賜わられたのであった。

1）の65536色モードでは，C00000，C00001$_H$のいずれにもメモリがあり，書き込まれた1ワードのデータのすべてが有効になる。

2）の256色モードではC00001$_H$にのみメモリがあり，C00000$_H$に対しても1ワードのデータの読み書きができるものの，実際には1バイト分のデータしか保持はされない。

3）の16色モードに至っては1バイトの半分しか実際にはメモリがなく，1ワードのデータの読み書きができるものの，半バイトしかデータを保持しない，というのである。かくして吾輩は，512Kバイトのメモリを非常に効率よく使うことが可能となり，図3のような豊富な画面モードを実現しているのである。

●禁断の24kHz

図4の100000$_H$でD1.wレジスタにセットしているデータを17に変更すれば，1024×424ドットの画面にすることができる。そして18にすれば，禁断の超高精彩表示を目にすることができるであろう。こいつの素晴らしさは我ながら惚れぼれするほどである。ただし，廉価版のディスプレイには24kHzモードがないので，ご愁傷さまと申し上げるしかない。

ディスプレイには個体差があるので，24kHzモードがあるからといって安心はできない。あるディスプレイではチラツキがひどくてほとんど文字を判別できない一方で，あるディスプレイではほとんど支障なく使えたりする。御仁の最近の関心事は，吾輩につなぐことのできる超長残光の24kHzディスプレイを入手する方法である。ご存じの方は教えてやっていただきたい。

IOCSコール10$_H$をここまで使ってきたやり方では，テキスト画面がクリアされ，グラフィック画面がOFFにされてしまう。画面を消去せずに，そして，グラフィックもOFFにすることなく画面モードを変更したいなら，図3の画面モードに対応する数に100$_H$を加えてD1.wにセットされたい。すなわち，

```
100000  move.w #16+$100,d1
```

のようにするわけである。これで，BASICのimg_scrn関数と同じ効果が得られる。試されたい。

よいこのSX-WINDOW講座（第5回）

# マウスカーソルを変更する

Nakamori Akira 中森 章

いつもなにげなく目にしているマウスカーソルですが，今回はこのマウスカーソルのしくみについて考えてみましょう。また，マウスカーソルのパターンを変えるためのプログラム例も用意しました。ぜひ参考にしてください。

前回の予告では，今回からはSX-WINDOWの各マネージャにスポットを当てて，そのマネージャごとの関数について解説する予定になっていました。が，よく考えてみると，行き当たりばったりにテーマを選んできたとはいえ，これまでもコントロールマン，ダイアログマン，……とマネージャ単位に話を進めてきていたのでした。今回から何が変わるかというと，実は変わりようがないのです。というわけで，これから先もこれまでと同様の方針で解説をしていくことにします。

さて，今回のテーマはマウスマンとアニメーションマンです。これらのマネージャはどちらもマウスカーソルの動作に関するものです。マウスカーソルは常にSX-WINDOWで作業をしている人の目に触れていますから，私たちにとってはもっとも馴染みの深いものといえるでしょう。そんな大事なマウスカーソルをシステムで初めから提供されているもので満足できるでしょうか。今回はマウスカーソルの形状を変更することに挑戦したいと思います。

## マウスに関するマネージャの概要

マウスマンとアニメーションマンはマウスの動作に関するマネージャです。マウスマンはマウスの移動量を認識して座標計算を行い，マウスカーソルを表示します。また，マウスボタンのクリックがあると，イベントマンを呼び出してイベントを発生します。それから，マウスカーソルのアニメーションは，マウスマンではなく，アニメーションマンによって行われます。

さて，これらのマネージャが有している具体的な機能は次のようになっています。

**1) カーソルの表示**
**2) カーソルの移動速度の変更**
**3) カーソルの形状の変更**
**4) 右利き/左利きマウスの切り替え**
**5) カーソルの前後左右の移動方向の反転**
**6) カーソルのアニメーション**

そして，これらの機能を使用するための関数が，マウスマンとアニメーションマンに用意されています。これらのマネージャで提供されている具体的な関数を表1に示しておきましょう。

ところで，表1を見るとわかりますが，マウスマンの機能のうち，4)と5)についてはマウスマンの関数という形では提供されていません。これらの機能を実現するためには特別な操作が必要です。これらについてはマウスを制御する構造体（MsRec）の内容を直接操作しなければなりません。ただし，MsRecという構造体の内容についてはSX-WINDOWのドキュメントに詳しい説明がありません。C言語用のヘッダファイルであるsxdef.hの中でもMsRecは，

typedef MsRec ｛｝ MsRec;

と名前が定義されているだけです。ドキュメントによると，ここらへんの構造は将来的に変更される可能性があるということです。そのためでしょうか。とはいえ，このままではMsRecは使いものにならないので，使用する場合には構造体をちゃんと定義してやらなければなりません。

**表1　マウスマン，アニメーションマンの関数**

| 関数名 | 機能 |
|---|---|
| MSInitCsr | カーソルレベルを0にして，カーソルをシステム標準のものにする |
| MSShowCsr | カーソルレベルを＋1し，0ならカーソルを表示する |
| MSHideCsr | カーソルレベルを－1し，カーソルを消す |
| MSSetCsr | カーソルのパターンを変える |
| MSObscureCsr | カーソルレベルと無関係にカーソルを消す |
| MSShieldCsr | 四角形と重なる場合，レベルを－1してカーソルを消す |
| MSGetCurMsr | 現在のマウスレコードへのポインタを返す |
| MSMultiGet | マウスの移動係数を返す |
| MSMultiSet | マウスの移動係数を設定する |
| EXAnimStart | マウスパターンでカーソルのアニメーションを開始する |
| EXAnimEnd | カーソルのアニメーションを終了する |
| EXAnimTest | カーソルでアニメーションをしているかどうか調べる |

## マウスカーソルを表示する関数

SX-WINDOWのマウスカーソルにはカーソルレベルという概念があります。これは，マウスカーソルが表示できるかどうかを示すレベルです。つまり，カーソルレベルが0のときはマウスカーソルが表示されます。カーソルレベルが負のときはマウスカーソルは表示されません。また，正のカーソルレベルというものは存在しません。

MSShowCsr
MSHideCsr
MSShieldCsr

といった関数は，このマウスカーソルのレベルを操作することで，マウスカーソルを表示したり消したりするのです。

それでは，カーソルレベルはなんのためにあるのでしょう。これはマウスカーソルが消えているべきときに表示されてしまうことを防ぐためだと推測されます。たとえば，ある関数がマウスカーソルを消したなら，その関数を抜け出すときはマウスカーソルを表示するでしょう。当たり前のことのように思えますが，このような関数が入れ子で呼ばれた場合は制御が大変です。マウスカーソルをせっかく消したのに，ある関数を抜け出したときにマウスカーソルが勝手に表示されてしまうという事態が起こり得ます。

が，カーソルレベルという概念を使えばこのようなことはなくなります。つまり，カーソルレベルはマウスカーソルを消そう

とした回数（MSHideCsr関数やMSShieldCsr関数が呼ばれた回数）を示します。それと，同じ回数だけマウスカーソルを表示（MSShowCsr関数を呼ぶ）しようとしないとマウスカーソルが表示されないようになっているのです。これが，マウスカーソルを消そうとすると値が1減り，マウスカーソルを表示しようとすると値が1増加し，値が0のときのみマウスカーソルが実際に表示されるというカーソルレベルの仕組みなのです。図1にカーソルレベルとマウスカーソルの表示の関係を図示しておきましょう。

マウスマンの関数のなかには，カーソルレベルとは無関係にマウスカーソルを消す関数もあります。

MSObscureCsr

がその関数です。これは，キーボードによる文字の入力中などに，一時的にマウスカーソルを消しておきたい場合に使用します。この関数によって消されているマウスカーソルはマウスカーソルをその位置から移動させることによって再び表示されるようになります。

## マウスカーソルの移動速度を変更する関数

マウスカーソルの移動速度はマウスの移動係数によって決定されます。移動係数とは，マウスを少し移動させたとき，画面上のマウスカーソルを何ドット移動させるかを決めている定数です。このため，移動係数が大きければマウスカーソルは速く動くように見え，逆に移動係数が小さければマウスカーソルはゆっくり動くように見えるのです。なお，SX-WINDOWの初期状態では移動係数は256になっています。

マウスの移動係数に関係する関数は次の2つです。

MSMultiGet
MSMultiSet

MSMultiGet関数は移動係数を得るための関数，MSMultiSetは移動係数を変更するための関数です。MSMultiSet関数は変更前の移動係数を戻り値とします。

## マウスカーソルの形を変更する関数

マウスカーソルはSX-WINDOWではもっとも目につく対象ですから，マウスカーソルの形状を目的によって変化させることはアプリケーションプログラムとのインタフェイスを向上させるのに効果的です。たとえば，SX-WINDOWに付属してくる暁子.Xは，マウスカーソルがウィンドウ内にあるとその形状が骨つきの肉に変わるようになっています。また，オンラインソフトのSXecho.Xでは，文字列を入力するときに表示されるキャレットと同じような形状に変わるようになっています。これによって，暁子.Xでは「マウスカーソルを犬の目の前に持っていくのだな」とか，SXecho.Xでは「何か文字を入力するのだな」ということが一目でわかるようになっているのですね。

マウスカーソルの形状を変更するための関数は，

MSSetCsr

です。この関数はカーソルレコード（構造体TXcsr）へのハンドルを引数とし，現在のマウスカーソルの形状をそのカーソルレコードによって指定されたパターンで置き換えるのです。MSSetCsr関数に与えるハンドルは疑似ハンドルでかまいません。通常のハンドルは，メモリマンのMMChHdlNew関数によって領域を確保しますが，疑似ハンドルは領域を，MMChHdlNew関数を呼ばず，直接メモリ上に確保してよいのです。要は「(構造体TXcsrへの）ポインタへのポインタ」という形式さえ満たしていればなんでもよいのです。

カーソルレコードのTXcsrは次の構造をした構造体です。

```
typedef struct TXcsr {
    point_t          csrHot;
    unsigned shortcsrMask [16];
    unsigned shortcsrTXptn[4] [16];
} TXcsr;
```

この構造体はマウスカーソルの位置情報（csrHot），マスク情報（csrMask），パターン情報（csrTXptn）から構成されています。csrHotはパターンのどこがマウスカーソルのある位置（ものを指し示す点）になるかを指定します。csrMaskはマウスカーソルがある位置の背景の処理を指定するビットパターンです。これは16個の要素を持つ16ビット（short型）データの配列で，16×16ドットのパターンを表しています。なお，パターンのビット位置と配列要素の関係は図2のようになっています。このパターンの中の1の位置にcsrTXptnで指定されるマウスカーソルのパターンが埋め込まれて表示されると思ってよいでしょう。パターンの中の0の位置は背景がそのまま透けて見えるようになります（マウスカーソルのパターンのパレットによっては背景が反転)。最後のcsrTXptnはテキスト画面4プレーン分のマウスカーソルのパターンです。これは4つの16×16ドットのパターンを要素とする2次元配列です。

csrTXptn[0] [0]~csrTXptn[0] [15]
csrTXptn[1] [0]~csrTXptn[1] [15]
csrTXptn[2] [0]~csrTXptn[2] [15]
csrTXptn[3] [0]~csrTXptn[3] [15]

が，それぞれ，プレーン0からプレーン3に対応しています。そして，この4プレーン分のパターンで各点のパレットコードを指定します。たとえば，ある点におけるプレーン0からプレーン3までのドットの値が0，0，0，1となっていると，その点のパレットコードは2進数で1000，つまり8となります。パレットコードと表示される色の関係はSX-WINDOWのドキュメントを見てください。なお，各プレーンに属

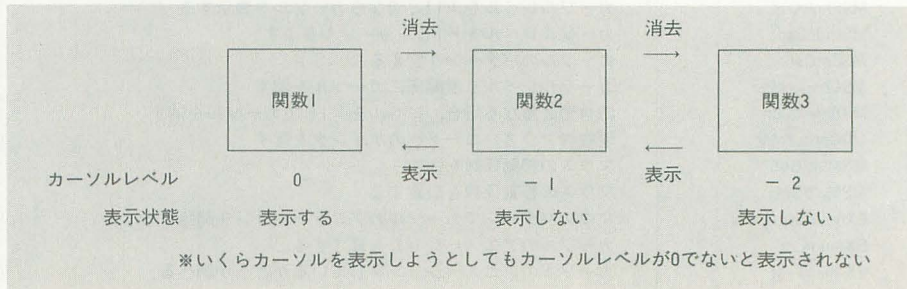
### 今月のバグ出し

sxlib.h（正確にはsxlib.hでインクルードされているsxdef.h）の中にMsRecという構造体の実体が定義されていません。これではマウスレコードを操作することができませんから，sxdef.hの中のMsRecの定義を次のように書き換えてください。

```
typedef struct MsRec {
    unsigned int     msBitMap;
    unsigned short   msMulti;
    unsigned char    msRvsSwitch;
    unsigned char    msRvsForBack;
    unsigned char    msRvsLeftRight;
} MsRec;
```

マウスレコードはこれよりも多くのフィールドを持っているようです（ほとんどがシステム内で使用するワークエリア）が，とりあえずこれだけ定義しておけば不便はないでしょう。

図1　カーソルレベルとカーソル表示の関係

する16×16ドットのパターンはcsrMaskと同様の16個の16ビット（short型）データの形式をしています。

ところで，XCには2進定数の表現が許されていますから，それを利用すれば上のマスク情報や1プレーン分のパターン情報は，

```
 0b1000000000000000
,0b1100000000000000
,0b1110000000000000
,0b1111000000000000
,0b1111100000000000
,0b1111110000000000
,0b1111111000000000
,0b1111111100000000
,0b1111111110000000
,0b1111111111000000
,0b1111111111100000
,0b1111111111110000
,0b1111000000000000
,0b1110000000000000
,0b1100000000000000
,0b1000000000000000
```

などと，見たままのイメージで表現することができます[1]。これはSX-WINDOWの標準的なマウスカーソルのパターンのプレーン3のイメージです。

さて，マウスカーソルの形状はMSSetCsr関数によって変更することができますが，これはマウスカーソルの形状を永久に変更するものではありません。新たなアプリケーションを起動したりすると，システム標準の形状に戻ってしまいます。これによって，マウスカーソルの形状の変更が「あるウィンドウがアクティブなあいだだけそのウィンドウに特有の操作を行うために提供されている機能である」と理解することができます。SX-WINDOWのシステムのマウスカーソルの形状が気に入らないからといって，標準的なマウスカーソルの形状を変更するという性格のものではないようです。

1）以前おまけディスクで提供したGCCのVer1.36.01でも2進定数をサポートしたが，配列などの初期値として2進定数を使用すると正しくコンパイルできない。また，XCのver.1.0では構造体の初期化を間違えることがある。したがって，マウスカーソルのパターンなどを2進定数で記述してコンパイルするためには実質的にXCのver.2.0を使用しなければならない。

## 左利きマウスと右利きマウスを使う

SX-WINDOWでサポートされているマウスは右利きマウスです。これはマウスを右手で操作することが前提となっています。つまり右手でマウスを握ったとき，頻繁に押すことになる左ボタンが右手の人指し指に近くなるようになっているのです。右利きマウスでは，マウスカーソルの向きも右手で物を指差したときと違和感がないように左向きになっています。

ところが，これはまさに右利きの人のための設計であって左利きの人には使いにくいものになっているはずです。そこで，SX-WINDOWでは左利きの人のために左利きマウスもサポートされています。左利きマウスでは，マウスの左ボタンと右ボタンの働きが逆になります。また，マウスカーソルの向きも，左向きから右向きに変更になります。

機能的にはSX-WINDOWでサポートされている左利きマウスですが，マウスマンの関数では右利きマウスと左利きマウスを切り換える機能を持つものはありません。ただ，システムを起動するときにマウスの左右のボタンを同時に押していると，左利きマウスでシステムが起動するという機能があるのみです。

ただし，アプリケーションプログラムで右利きマウスと左利きマウスを切り替えることができないかというと，そうではありません。マウスの動作を管理するマウスレコードを直接操作してやれば可能です。マウスレコード（構造体MsRec）へのポインタはMSGetCurMsr関数で獲得することができますから，その構造体のmsRvsSwitchフィールドの値を変更してやればよいのです[2]。そのフィールドの値が，

0x00なら　右利きマウス

0xffなら　左利きマウス

です。ただし，このフィールドの値を変更しただけでは，マウスの左ボタンと右ボタンの機能が入れ替わるだけで，マウスカーソルの向きは変わりません。マウスカーソルの向きを変更するためには，msRvsSwitchフィールドの値を変更したあとにMSI

左手用マウスカーソル

**図2　パターンのビット位置と配列要素の対応**

| 16×16ドットのビットイメージ | | 16個のshort型データ（2進） | | 16進 | | 配列要素 |
|---|---|---|---|---|---|---|
| ●○○○○○○○○○○○○○○○ | → | 1000000000000000 | → | 8000 | ←→ | csrMask [0] |
| ●●○○○○○○○○○○○○○○ | → | 1100000000000000 | → | C000 | ←→ | csrMask [1] |
| ●●●○○○○○○○○○○○○○ | → | 1110000000000000 | → | E000 | ←→ | csrMask [2] |
| ●●●●○○○○○○○○○○○○ | → | 1111000000000000 | → | F000 | ←→ | csrMask [3] |
| ●●●●●○○○○○○○○○○○ | → | 1111100000000000 | → | F800 | ←→ | csrMask [4] |
| ●●●●●●○○○○○○○○○○ | → | 1111110000000000 | → | FC00 | ←→ | csrMask [5] |
| ●●●●●●●○○○○○○○○○ | → | 1111111000000000 | → | FE00 | ←→ | csrMask [6] |
| ●●●●●●●●○○○○○○○○ | → | 1111111100000000 | → | FF00 | ←→ | csrMask [7] |
| ●●●●●●●●●○○○○○○○ | → | 1111111110000000 | → | FF80 | ←→ | csrMask [8] |
| ●●●●●●●●●●○○○○○○ | → | 1111111111000000 | → | FFC0 | ←→ | csrMask [9] |
| ●●●●●●●●●●●○○○○○ | → | 1111111111100000 | → | FFE0 | ←→ | csrMask [10] |
| ●●●●●●●●●●●●○○○○ | → | 1111111111110000 | → | FFF0 | ←→ | csrMask [11] |
| ●●●●○○○○○○○○○○○○ | → | 1111000000000000 | → | F000 | ←→ | csrMask [12] |
| ●●●○○○○○○○○○○○○○ | → | 1110000000000000 | → | E000 | ←→ | csrMask [13] |
| ●●○○○○○○○○○○○○○○ | → | 1100000000000000 | → | C000 | ←→ | csrMask [14] |
| ●○○○○○○○○○○○○○○○ | → | 1000000000000000 | → | 8000 | ←→ | csrMask [15] |

※パターンの1行がひとつのshort型データに対応する

nitCsr関数でマウスカーソルを初期化する必要があります。

なお，msRvsSwitchフィールドの値を変更するときは注意が必要です。マウスレコードの実体はシステム領域にあるので，C言語の代入文で値を参照したり変更することはできません。IOCSライブラリのB_BPEEK関数やB_BPOKE関数を利用する必要があります。このため，そのような処理を行う場合はコンパイル時にIOCSライブラリをリンクしなければなりません[3]。

---

2) 先に述べたように，sxlib.hの中ではMsRecの実体の定義がないので，まずそれを定義しなければならない（今月のバグ出しのコーナーを参照のこと）。

3) コンパイル時にIOCSライブラリをリンクするオプションをつけること。XCでは/Y，GCCでは-liocsである。

## マウスカーソルの移動方向を反転する

マウスを右に移動するとマウスカーソルは右に移動します。マウスを左に移動するとマウスカーソルは左に移動します。マウスを上に移動するとマウスカーソルは上に移動します。マウスを下に移動するとマウスカーソルは下に移動します。普段は気にもしないマウスの移動方向とマウスカーソルの関係ですが，SX-WINDOWではこの関係を変更することができます。この機能はどういう目的で使用するのかよくわかりません。ジョイスティックを使用するゲームには，リバース機能と称してスティックの上下の動きと画面の物体の動きの方向を逆転する機能があります。それと同様にゲームなどで利用する機能なのでしょうか。

## ダイアログの補足

本誌7月号の連載（第3回）で，ダイアログ実行中はほかの処理が停止するので，一定時間が経過したあとにダイアログウィンドウをクローズするような処理が必要なら，ダイアログマンを使用せず通常のウィンドウでダイアログを実現しなければならない，という説明をしました。それに対し，大阪府の嶋田裕文さんより，そのような処理はフィルタ関数をうまく利用することで可能ではないか，というお便りをもらいました。ヌルイベントもフィルタ関数に通知されているというのがその理由だそうです（デバッガで調べたのだそうな）。

それについて調べたので報告します。結果として，フィルタ関数にヌルイベントの発生が通知されることはないようです。しかし，フィルタ関数は一定時間ごとに呼び出されている（ヌルイベントが発生するごとに呼び出される？）ようなので，嶋田さんがおっしゃるような処理は可能です。リスト5に時刻を表示しながらダイアログを行うためのフィルタ関数の例を示します。前回で示したスケルトンプログラムや今回のサンプルプログラムなどに組み込んで動かしてみてください。

嶋田さん，ありがとうございました。皆さんも有用な情報がありましたらぜひ教えてくださいね。

**リスト5 フィルタ関数でこんなこともできる**

```
 1: /*
 2:         フィルター関数はこんなこともできる
 3:     (一部分なのでこのままでは動作しません)
 4: */
 5: time_t oldtime=0;
 6: time_t newtime;
 7: char chartime[10];
 8: rect timerect={8,128-20,64,128-6};
 9: point_t timept;
10:
11: MyFilter(Dialog,ev)
12: dialog *Dialog;
13: event  *ev;
14: {
15:     point_t okbtn;
16:
17:
18:     newtime=time(NULL);
19:     if(newtime!=oldtime){
20:         oldtime=newtime;
21:         strncpy(chartime,ctime(&newtime)+11,8);
22:         chartime[8]=NULL;
23:         timept.p.x=12;
24:         timept.p.y=128-20;
25:         GMSetGraph(Dialog);
26:         GMShadowRect(&timerect);
27:         GMShadowStrZ(chartime,timept);
28:     }
29:
30:     if( ev->eWhat == E_KEYDOWN ){
31:         if((short)(ev->eWhom)==13){
32:             okbtn.p.x=384+128-10;
33:             okbtn.p.y=256+64-20-10;
34:             ev->eWhere=okbtn;
35:             ev->eWhat =E_MSLDOWN;
36:         }
37:     }
38:     return 0;
39: }
```

それはともかく，マウスレコード（MsRec）のmsRvsForBackフィールドが上下方向，msRvsLeftRightフィールドが左右方向を決定するフィールドになっているので，その値を変更することでマウスカーソルの移動方向を逆転することができます。フィールドの値が，

　0x00　なら　通常の方向
　0xff　なら　通常と逆の方向

です。右利きマウス，左利きマウスの切り替えのときと同じく，これらのフィールドの参照・変更はIOCSライブラリのB_BPEEK関数，B_BPOKE関数を使用します。

## マウスカーソルでアニメーションする

SX-WINDOWではマウスカーソルでアニメーションを行うことができます。この機能はファイルのサーチなどの時間がかかる処理を行っているとき，待ち状態にあることを示すためによく利用されます（つまり，システムがハングアップしたのではないことの意思表示？）。砂時計の砂が落ちているアニメーションはいろいろなウィンドウシステムでよく見掛けます。SX-WINDOWでは待ち状態専用にカーソルを踏み切りの形状に変えてアニメーションをする機能（ダイアログマンの関数であるDMWaitOpen）を持っていますが，それを一般化した機能と思ってよいでしょう。

マウスカーソルのアニメーションは次の3つの関数によって行います。

　EXAnimStart
　EXAnimEnd
　EXAnimTest

上から順に，アニメーションを始める関数，終了する関数，アニメーションを行っているかテストする関数です。使用方法は難しくありませんが，EXAnimStartの引数は注意が必要です。EXAnimStartの第3引数で実行するアニメーションのパターンを与えます。これはマウスカーソルレコード（TXcsr）へのハンドルを要素とする配列へのポインタ（配列名）となります。要素のハンドルは疑似ハンドルでよいので，メモリマンのMMChHdlNew関数を呼ばず，パターンを直接メモリ上に確保することができます。データ構造が複雑なので，間違えると正しくアニメーションが行われません。このパターンの配列のデータ構造は，

　TXcsr pat1 = {…}；
　TXcsr* pat1Ptr = &pat1；

によってひとつのパターンが定義されているとき，そのパターンへのポインタ（pat1

Ptr)へのポインタ(&pat1Ptr)を要素とする配列になるのです。

ところで，アニメーションによるマウスカーソルのパターンの変更はかなり「強い」変更です。MSSetCsr関数でパターンを変更した場合は，別のアプリケーションの起動ですぐ元に戻ってしまいます。しかし，アニメーションを行っているときは，そのアニメーションをやめない限り，マウスカーソルのパターンを変更することはできません[4]。したがって，マウスカーソルのパターンを変更するプログラムでマウスカーソルが正しく変化しなくなってしまうので，アニメーションを開始したアプリケーションのウィンドウがインアクティブになるときは，アニメーションを停止しなければならないでしょう。マウスカーソルのアニメーションは時と場合を選んで効果的に使用しなければなりません。

---

4) これを逆に利用してパターン数が1のアニメーションを行えば，システムに標準のマウスカーソルが表示されないようにする(マウスカーソルの形状を永久に変更する)こともできるが……。

## プログラムの例

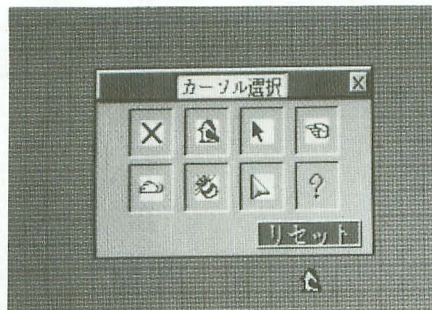
マウスカーソルの形状を変更することを主体としたサンプルプログラムをリスト1～4に示します。このプログラムは，

- リスト1　メインプログラム
- リスト2　マウスの動作を変更するためのダイアログ
- リスト3　MSSetCsrで変更するマウスカーソルのパターン(TXcsr構造体)の定義
- リスト4　アニメーションするためのマウスカーソルのパターンの定義

という構成になっています。

このプログラムはウィンドウ上に8つの領域があって(ウィンドウがアクティブのとき)，その領域内にマウスカーソルを移動すると，マウスカーソルをその領域に示してある形状に変化させるものです。マウスカーソルをその領域の外に出すとマウスカーソルの形状は元に戻ります(アイドルイベント時にマウスカーソルの位置を調べて形状を変更したり元に戻したりしています)。そして，マウスカーソルが変化しているときにマウスの左ボタンを押すと，マウスカーソルの形状を固定します(こちらは左ボタンのダウンイベント時に処理をしている)。つまり，領域の外にマウスカーソルを移動しても形状が元に戻りません。「?」のところでマウスの左ボタンを押すとマウスカーソルのアニメーションが始まります。また，ポップアップメニューの3番目の項目を選択するとマウスに関する環境設定のダイアログが出ます。これによって，左利きマウス，移動方向の逆転，移動係数の変化，カーソルの一時消去を体験できるようになっています。プログラム的にはマウスマンやアニメーションマンの関数を呼び出しているだけですので，特に説明はいらないでしょう(必要なところにはリストにコメントをつけてありますし)。

なお，リスト3とリスト4で定義してあるマウスカーソルのパターンのうち，handcs，mousecs，apat1～apat5は梁山泊ネットのCANDY氏作成のSXmscsr.x(マウスカーソルの形状を変えるプログラム)に付属していたパターンを参考にしています。

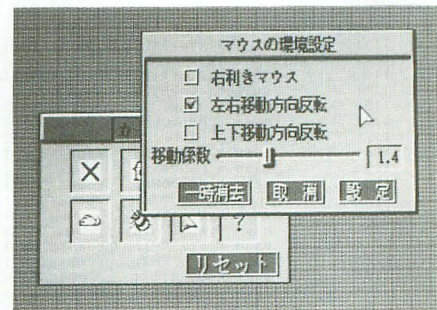
なお，リスト1～リスト4のコンパイルは，この連載の第1回，第2回目で紹介したバッチファイルなどで一括してコンパイルしてください(IOCSライブラリのオプションを忘れないように)。ただし，XCのver.2.0以外のコンパイラを使用するときは，リスト3とリスト4はXCのver.2.0でいったんコンパイルしたあとで，そのファイル(.sファイル，または.oファイル)をコンパイル用のバッチファイルに渡すようにしてください。

## おわりに

今回は，マウスカーソルに関していろいろな設定変更を行うプログラムを作ってみました。マウスのカーソルレベルを操作してカーソルを表示したり消去したりする例はありませんが，これは各自の自習とします。

今回のサンプルプログラムはあまり実用的なものではありませんが(環境ソフトとしてもイマイチ)，マウスカーソルの形状を変えたり，マウスレコードの内容を変更したりすることによってどのような効果が生まれるかを体験しておけば，これから先に自分で作るアプリケーションのなかでちょっとしたマウスカーソルの設定変更を隠し味として使用することが可能になると思います。一味違ったプログラムを作って他人を感心させてしまいましょう。


≪参考文献≫

1)吉沢正敏，SX-WINDOWプロラミング，ソフトバンク，1991年.


起動メニュー

マウスカーソルの設定変更

### リスト1

```
 1: /*
 2:
 3:      マウスカーソルの選択(スケルトン Ver.2.0 より)
 4:
 5: 【改版履歴】
 6:
 7:      Jul.11,1991 新規作成
 8:
 9:          (C) 中森 章, Jul.11, 1991
10: */
11: #include <stdio.h>
12: #define __POINT_T   /* point_t 型を使う */
13: #include <sxlib.h>
14: #define FALSE   0
15: #define TRUE    ~FALSE
16:
17: #define EXT_GETWINSIZE  /* getWinSize() は外部で定義 */
18: #undef  EXT_GETWINSIZE  /* してはいない              */
19: #define ICON_WIDTH  200
20: /*
21:      ここでウィンドウに関する定数を設定
22: */
23: #define WDEFID        WI_STD
24: #define WINOPT        0
25: #define WINWIDTH      200
26: #define WINHIGHT      100
27: #define WINTITLE      "¥014カーソル選択"
28: #define EVENTMASK     EM_EVERY
29:
30: #define MDEFID        0    /* タイトルなしメニュー */
31: #define MNENABLE      0xffffffff
32: #define MNITEMS       2
33: #define MNILIST       "¥0¥0¥025カーソル選択について ¥0¥0¥013アイコン…
"
34: #define MNTITLE       "¥014メニューだよ"
35:
36: #define CTRLID        CI_STDBTN    /* 標準ボタン   */
37: #define CTRLMIN       0          /* 最小値   */
38: #define CTRLMAX       1          /* 最大値   */
39: #define CTRLVAL       0          /* 初期値   */
```

```
40: #define CTRLTITLE    "¥010リセット"  /* タイトル */
41: #define CTRLLEFT     WINWIDTH-84 /* 左上X座標   */
42: #define CTRLTOP      WINHIGHT-22 /* 左上Y座標   */
43: #define CTRLRIGHT    CTRLLEFT+76 /* 右下X座標   */
44: #define CTRLBOTTOM   CTRLTOP +18 /* 右下Y座標   */
45: /*
46:         ここは定数から計算される定数
47: */
48: #define WINOPTL       ( WINOPT & 0xf )
49: #define WINDEFID      ( WDEFID << 4 | WINOPTL )
50:
51: rect    getWinSize(int,int);
52:
53: window  *winPtr;
54: rect    winSize;
55: event   eventRec;
56: int activeFlag;
57:
58: int     ctrlFlag;   /* コントロールがあるかないか */
59: int menuFlag;    /* メニューがあるかないか */
60: int iconFlag;    /* アイコンになっているかどうか */
61: int lastWhen;    /* ダブルクリックの判定用 */
62: point_t oldWinSize; /* アイコン時のウィンドウサイズ記憶用 */
63:
64: menu    **menuHdl;
65:
66: menu theMenu = {
67:     0,0,0,0,MNENABLE,0,{MNITEMS-1,MNILIST}
68: };
69:
70: control **ctrlHdl;  /* コントロールへのハンドル */
71: rect    ctrlSize;   /* コントロールの大きさ      */
72: int ctrlValue;  /* コントロールの値        */
73:
74: control **ctrlSelHdl;    /* 選択されたコントロールを格納 */
75:
76: rect    updRect={0,0,WINWIDTH,WINHIGHT};
77: /*
78:         疑似ハンドルを作る
79: */
80: extern TXcsr cross_cs,peng_cs,vsh_cs,hand_cs,
81:              mouse_cs,hudson_cs,shadow_cs;
82:
83: TXcsr *csrMenu[]={      /* マウスカーソル変更用 */
84:     &cross_cs,
85:     &peng_cs,
86:     &vsh_cs,
87:     &hand_cs,
88:     &mouse_cs,
89:     &hudson_cs,
90:     &shadow_cs,
91:     NULL
92: };
93:
94: extern TXcsr **msapat[]; /* アニメーション用 */
95:
96: int msCsrPat=-1;
97:
98: struct {
99:     rect bounds;
100:    unsigned short TXptn[4][16];
101: }
102: RectImage={
103:     {0,0,16,16},
104:     {0}
105: };
106:
107: TXcsr **CsrHdl;
108:
109: main()
110: {
111:     if( SX_init()==FALSE ) OpenError() ;
112:     while( 1 ){
113:         TSEventAvail(EVENTMASK,&eventRec);
114:         switch( eventRec.eWhat ){
115:         case E_IDLE:      procIDLE();      break;
116:         case E_MSLDOWN:   procMSLDOWN();   break;
117:         case E_MSLUP:     procMSLUP();     break;
118:         case E_MSRDOWN:   procMSRDOWN();   break;
119:         case E_MSRUP:     procMSRUP();     break;
120:         case E_KEYDOWN:   procKEYDOWN();   break;
121:         case E_KEYUP:     procKEYUP();     break;
122:         case E_UPDATE:    procUPDATE();    break;
123:         case E_ACTIVATE:  procACTIVATE();break;
124:         case E_SYSTEM1:   procSYSTEM();    break;
125:         case E_SYSTEM2:   procSYSTEM();    break;
126:         case E_USER1:     procUSER();      break;
127:         case E_USER2:     procUSER();      break;
128:         }
129:     }
130: }
131:
132: #ifndef EXT_GETWINSIZE
133: rect
134: getWinSize(xmin,ymin)
135: int xmin,ymin;
136: {
137:     rect    r;
138:     *(int *)&r.left = TSGetWindowPos();
139:     r.right = r.left+xmin;
140:     r.bottom= r.top +ymin;
141:     return  r;
142: }
143: #endif
144: /*
145:         ウィンドウ上でマウスカーソルのある位置を調べる
146:         (座標はローカル座標)
147: */
148: checkCsrPos(p)
149: point_t p;
150: {
151:     int x,y;
152:
153:     x=p.p.x;
154:     y=p.p.y;
155:     if(y>=4 && y<=36){
156:         if(x>=24 && x<=56)          return(0);
157:         else if(x>=64 && x<=96)     return(1);
158:         else if(x>=104 && x<=136) return(2);
159:         else if(x>=144 && x<=176) return(3);
160:         else return(-1);
161:     }
162:     else if( y>=40 && y<=72){
163:         if(x>=24 && x<=56)          return(4);
164:         else if(x>=64 && x<=96)     return(5);
165:         else if(x>=104 && x<=136) return(6);
166:         else if(x>=144 && x<=176) return(7);
167:         else return(-1);
168:     }
169:     else
170:         return(-1);
171: }
172: /*
173:     ウィンドウを描き直す
174: */
175: drawWindow()
176: {
177:     int i,j;
178:     rect r;
179:     point_t p;
180:     int  ptn=0;
181:
182:     GMSetGraph((graph*)winPtr);
183:     GMPenMode(G_BACK|G_PSET);
184:     GMFillRect(&updRect);
185:     GMPenMode(G_FORE|G_PSET);
186:     GMFontKind(G_ROM16);
187:     GMAPage(7);
188:     for(j=4;j<100-36;j+=36)
189:         for(i=24;i<200-40;i+=40){
190:             r.left=i;
191:             r.top=j;
192:             r.right=i+32;
193:             r.bottom=j+32;
194:             GMShadowRect(&r);
195:             p.p.x=i+8;
196:             p.p.y=j+8;
197:             if(csrMenu[ptn]==NULL){ /* ?マークを表示 */
198:                 GMShadowStrZ("?",p);
199:             }
200:             else{ /* マウスカーソルのパターンを表示 */
201:                 memcpy( &RectImage.TXptn,
202:                     &(csrMenu[ptn]->csrTXptn),
203:                     4*16*sizeof(short));
204:                 GMPutRImg((rectImg*)&RectImage,p);
205:             }
206:             ptn++;
207:         }
208:     GMAPage(3);
209: }
210:
211: SX_init()
212: {
213:     task    taskBuf;
214:     char    BUF[100];
215:
216:     TSGetTdb(&taskBuf, -1);
217:     if( (TSTakeParam(&taskBuf.command,&winSize,NULL,0,NULL
NULL)&1)==0 ){
218:         iconFlag=FALSE;
219:         winSize=getWinSize(WINWIDTH,WINHIGHT);
220:         oldWinSize.p.x=winSize.right-winSize.left;
221:         oldWinSize.p.y=winSize.bottom-winSize.top;
222:     }
223:     else{
224:         recovSize();
225:         if(winSize.top==winSize.bottom){
226:             iconFlag=TRUE;
227:         }
228:         else{
229:             iconFlag=FALSE;
230:             oldWinSize.p.x=winSize.right-winSize.left;
231:             oldWinSize.p.y=winSize.bottom-winSize.top;
232:         }
233:     }
234:     winPtr=WMOpen(NULL,&winSize,WINTITLE,TRUE,WINDEFID,(wi
ndow *)-1,TRUE,TSGetId());
235:     if( winPtr == NULL ) return( FALSE );
236:     winPtr->wOption = WINOPT;
237:     activeFlag=FALSE;
238:     msCsrPat=-1;
239:     lastWhen=-1;
240:     ctrlFlag  = CtrlPrepare();/* コントロールが不要なら ctrlFlag=
FALSE */
241:     menuFlag  = MenuPrepare();/* メニューが不要なら       menuFlag
=FALSE */
242:     drawGrowBox();
243:     return( TRUE );
244: }
245:
246: SX_term()
247: {
248:     if(EXAnimTest()) EXAnimEnd();
249:     MSInitCsr();
250:
251:     if( ctrlFlag ) CtrlDispose();
252:     if( menuFlag ) MenuDispose();
253:     WMDispose( winPtr );
254:     exit();
255: }
256:
257: drawGrowBox()
258: {
259:     GMSetGraph( winPtr );
260:     WMDrawGBox( winPtr );
261: }
262:
263: CtrlPrepare()
264: {
265:     ctrlSize.left   =CTRLLEFT;
266:     ctrlSize.top    =CTRLTOP;
267:     ctrlSize.right  =CTRLRIGHT;
```

▶ 8月15日に9月号があった。お盆ってすごいねえ。　　中村　直哉(18)北海道

```
268:     ctrlSize.bottom=CTRLBOTTOM;
269:     ctrlHdl=CMOpen(winPtr,&ctrlSize,CTRLTITLE,TRUE,CTRLVAL
,CTRLMIN,CTRLMAX,
270:             (CTRLID<<4), 0 );
271:     if(ctrlHdl==NULL){
272:         DMError(0x101,"コントロールがオープンできません");
273:         return ( FALSE );
274:     }
275:     return( TRUE );
276: }
277:
278: CtrlDispose()
279: {
280:     CMDispose(ctrlHdl);
281:     return( TRUE );
282: }
283:
284: MenuPrepare()
285: {
286:     menuHdl=(menu**)MMChHdlNew( sizeof(theMenu) );
287:     if( menuHdl == NULL ) return ( FALSE );
288:     memcpy(*menuHdl,&theMenu,sizeof(theMenu));
289:     (*menuHdl)->mProc=RMRscGet(('M'<<24)|('D'<<16)|('E'<<8
)|'F',MDEFID);
290:     if( (int)((*menuHdl)->mProc)<=0 ){
291:         MMHdlDispose(menuHdl);
292:         return( FALSE );
293:     }
294: #if MDEFID==1
295:     (*menuHdl)->mHandle=MNTITLE;
296: #endif
297:     return( TRUE );
298: }
299:
300: MenuDispose()
301: {
302:     MMHdlDispose(menuHdl);
303:     return( TRUE );
304: }
305: /*
306:     マウスカーソルが特定の領域にある間は
307:     マウスカーソルを変更する。
308:     その領域から出ると、マウスカーソルを
309:     もとに戻す。
310: */
311: procIDLE()
312: {
313:     int part;
314:     static int pattern=-1;
315:
316:     if( eventRec.eWhom != winPtr ) return( FALSE );
317:     if( activeFlag == FALSE ) return( FALSE );
318:     GMSetGraph( winPtr );
319:     part=checkCsrPos(GMGlobalToLocal(eventRec.eWhere));
320:     if( part==pattern ) return( TRUE );
321:     if(part>=0 && part<=6){ /* パターン領域内 */
322:         if(EXAnimTest() && msCsrPat==7) EXAnimEnd();
323:         CsrHdl=csrMenu[part];
324:         MSSetCsr(&CsrHdl);
325:     }
326:     else if(part==7){         /* パターン領域内 */
327:         MSSetCsr(*msapat);
328:     }
329:     else {  /* パターン領域外(元のパターンに) */
330:         if(msCsrPat>=0 && msCsrPat<=6){
331:             CsrHdl=csrMenu[msCsrPat];
332:             MSSetCsr(&CsrHdl);
333:         }
334:         else if(msCsrPat==7){
335:             if(!EXAnimTest()) EXAnimStart(5,10,msapat);
336:         }
337:         else
338:             MSInitCsr();
339:     }
340:     pattern=part;
341:     return( TRUE );
342: }
343: /*
344:     マウスカーソルが特定領域内にあるときに
345:     マウスの左ボタンを押すと、そのパターン
346:     にマウスカーソルを変更する
347: */
348: procMSLDOWN()
349: {
350:     int part;
351:
352:     if( eventRec.eWhom != winPtr ) return( FALSE );
353:     if( activeFlag == FALSE ){
354:         WMSelect( winPtr );
355:         activeFlag = TRUE;
356:         if( EMLStill() == 0) goto checkDClick;
357:     }
358:     switch( SXCallWindM(winPtr,&eventRec) ){
359:     case W_INCLOSE:
360:         SX_term(); break;
361:     case W_INGROW:
362:     case W_INZMOUT:
363:     case W_INZMIN:
364:         GMClipRect(&winPtr->wGraph.grRect);
365:         break;
366:     }
367:     part=SXCallCtrlM(winPtr,&eventRec,NULL,NULL,NULL,&ctrl
SelHdl);
368:     if(ctrlHdl==ctrlSelHdl){ /* マウスカーソルのリセット */
369:         if(EXAnimTest()) EXAnimEnd();
370:         MSInitCsr();
371:         msCsrPat=-1;
372:     }
373:     else{
374:         part=checkCsrPos(GMGlobalToLocal(eventRec.eWhere))
;
375:         if(part>=0){
376:             msCsrPat=part;
377:             if(part==7){
378:                 if(!EXAnimTest()) EXAnimStart(5,10,msapat)
;
379:             }
380:             else{
381:                 if(EXAnimTest()) EXAnimEnd();
382:                 CsrHdl=csrMenu[part];
383:                 MSSetCsr(&CsrHdl);
384:             }
385:         }
386:     }
387:
388: checkDClick:
389:     if(iconFlag==TRUE){ /* アイコンになっている */
390:         if(lastWhen==(-1))
391:             lastWhen=eventRec.eWhen;
392:         else{
393:             if((eventRec.eWhen-lastWhen)<EMDClickGet()){
394:                 lastWhen=-1;
395:                 toWindow();
396:             }
397:             else
398:                 lastWhen=eventRec.eWhen;
399:         }
400:     }
401:     TSGetEvent(EVENTMASK,&eventRec);
402:     return( TRUE );
403: }
404:
405: toWindow()
406: {
407:     iconFlag=FALSE;
408:     WMSize(winPtr,oldWinSize,-1);
409: }
410:
411: toIcon()
412: {
413:     rect    r;
414:     point_t p;
415:
416:     iconFlag=TRUE;
417:     r=winPtr->wGraph.grRect;
418:     oldWinSize.p.x=r.right-r.left;
419:     oldWinSize.p.y=r.bottom-r.top;
420:     p.p.x=ICON_WIDTH;
421:     p.p.y=0;
422:     WMSize(winPtr,p,-1);
423: }
424:
425: procMSLUP()
426: {
427:     return( FALSE );
428: }
429:
430: procMSRDOWN()
431: {
432:     int item;
433:
434:     if( eventRec.eWhom != winPtr ) return( FALSE );
435:     GMSetGraph( winPtr );
436:     if( activeFlag == FALSE ) return( FALSE );
437:     item=MNSelect(menuHdl,eventRec.eWhere);
438:     TSGetEvent(EVENTMASK,&eventRec);
439:     switch(item){
440:     case 1:
441:         doDialog(); break;
442:     case 2:
443:         if(iconFlag==TRUE)
444:             toWindow();
445:         else
446:             toIcon();
447:         break;
448:     }
449:     return( TRUE );
450: }
451:
452: procMSRUP()
453: {
454:     return( FALSE );
455: }
456:
457: procKEYDOWN()
458: {
459:     return( FALSE );
460: }
461:
462: procKEYUP()
463: {
464:     return( FALSE );
465: }
466:
467: procUPDATE()
468: {
469:     if( eventRec.eWhom != winPtr ) return( FALSE );
470:     WMUpdate( winPtr );
471:     drawWindow();
472:     if( ctrlFlag ) CMDraw( winPtr );
473:     WMUpdtOver( winPtr );
474:     drawGrowBox();
475:     TSGetEvent(EVENTMASK,&eventRec);
476: }
477:
478: procACTIVATE()
479: {
480:     if( eventRec.eWhom == winPtr )    activeFlag = TRUE;
481:     else if( eventRec.eWhom != NULL ){
482:         if( activeFlag ) {
483:             activeFlag = FALSE;
484:             TSGetEvent(EVENTMASK,&eventRec);
485:         }
486:     }
487:     return( TRUE );
488: }
489:
490: procSYSTEM()
491: {
492:     switch( ((tsevent*)&eventRec)->what2 ){
493:     case CLOSEALL:
494:     case ENDTSK:
```

▶まさか移植されるとは思わなかった。飛翔鮫が本当に移植されるんですね。いやあ，うれしいです。何をさしおいても手に入れます(ちょっと興奮気味)。福岡　尚久(22)愛知県

```
495:          SX_term(); break;
496:      case WINDOWSELECT:
497:          WMSelect( winPtr ); break;
498:      case SAVE:
499:          saveSize();
500:          break;
501:      }
502: }
503:
504: procUSER()
505: {
506:      return( FALSE );
507: }
508:
509: OpenError()
510: {
511:      DMError(0x101,"ウィンドウがオープンできません");
512:      exit();
513: }
514:
515: /*************************************
516: * doDialog()  ダイアログを出す関数   *
517: *************************************/
518: /*
519:      ここでダイアログウィンドウに関する定数を設定
520: */
521: #define DWINDEFID    (38<<4)
522: #define DWINTITLE    "¥020ダイアログだよん"
523: /*
524:       アイテムリスト
525: */
526: typedef struct dlgItem2 {
527:      long           dlgIHdl;
528:      rect           dlgIBounds;
529:      unsigned char   dlgIType;
530:      unsigned char   dlgISize;
531:      unsigned char   dlgIData[32];
532: } dlgItem2;
533:
534: struct {
535:      short    itemNo;
536:      dlgItem2 dItem1;
537:      dlgItem2 dItem2;
538:      dlgItem2 dItem3;
539:      dlgItem2 dItem4;
540:      dlgItem2 dItem5;
541: } dItemList ={
542:      5-1,
543:     {
544:      0,
545:      {256-8-42,128-8-18,256-8,128-8},
546:      DT_STDBTN,
547:      32,
548:      "¥007 ＯＫ "
549:     },
550:     {
551:      0,
552:      {4,4,252,16},
553:      DT_STCTXT+DT_DISABL,
554:      32,
555:      "¥001¥024このプログラムは…"
556:     },
557:     {
558:      0,
559:      {4,40,252,52},
560:      DT_STCTXT+DT_DISABL,
561:      32,
562:      "¥001¥030マウスカーソルを変更する"
563:     },
564:     {
565:      0,
566:      {4,60,252,72},
567:      DT_STCTXT+DT_DISABL,
568:      32,
569:      "¥001¥026プログラムなんですって"
570:     },
571:     {
572:      0,
573:      {240-96,80,240,92},
574:      DT_STCTXT+DT_DISABL,
575:      32,
576:      "¥377¥020中森 章 1991.5.31"
577:     }
578: };
579:
580: /*
581:        ダイアログを開く位置（中央よりも少し上にしてある）
582: */
583: rect dlBounds={ 384-128,256-64-20,384+128,256+64-20 };
584: /*
585:        フィルター関数
586: */
587: MyFilter(Dialog,ev)
588: dialog *Dialog;
589: event  *ev;
590: {
591:      point_t okbtn;
592:
593:      if( ev->eWhat == E_KEYDOWN ){
594:          if((short)(ev->eWhom)==13){
595:              okbtn.p.x=384+128-10;
596:              okbtn.p.y=256+64-20-10;
597:              ev->eWhere=okbtn;
598:              ev->eWhat =E_MSLDOWN;
599:          }
600:      }
601:      return 0;
602: }
603:
604: doDialog()
605: {
606:      dialog   *dialogPtr;
607:      dlgIList **dIHdl;
608:      int  ditem;
609:
610:      dIHdl=(dialog**)MMChHdlNew( sizeof(dItemList) );
611:      if( dIHdl == NULL ){
612:          DMError(0x101,"領域確保に失敗しました。");
613:          return ( FALSE );
614:      }
615:      memcpy(*dIHdl,&dItemList,sizeof(dItemList));
616:      dialogPtr=DMOpen(NULL,&dlBounds,DWINTITLE,TRUE,DWINDEF
ID,
617:              (window *)-1,TRUE,TSGetID(),dIHdl);
618:      if( dialogPtr == NULL ){
619:          MMHdlDispose(dIHdl);
620:          DMError(0x101,"ウインドウがオープンできません。");
621:          return( FALSE );
622:      }
623:      DMBeep(2);
624:      ditem=DMControl((void*)MyFilter );
625:      DMDispose(dialogPtr);
626:      MMHdlDispose(dIHdl);
627:
628: }
629:
630: saveSize()
631: {
632:      task    taskBuf;
633:      int len;
634:      int i;
635:      char    BUF[256];
636:
637:      sprintf(BUF," -S%d,%d ",
638:          oldWinSize.p.x,oldWinSize.p.y);
639:      len=strlen(BUF);
640:      if(len>255) len=255;
641:      TSGetTdb(&taskBuf, -1);
642:      for(i=0;i<len;i++)
643:          taskBuf.command.Lstr[i]=BUF[i];
644:      taskBuf.command.length=len;
645:      TSSetTdb(&taskBuf, -1);
646: }
647:
648: recovSize()
649: {
650:      task    taskBuf;
651:      int x,y;
652:      char    BUF[256];
653:
654:      TSGetTdb(&taskBuf, -1);
655:      sscanf(taskBuf.command.Lstr," -S%d,%d %s",
656:          &x,&y,BUF);
657:      oldWinSize.p.x=x;
658:      oldWinSize.p.y=y;
659: }
```

## リスト2

```
 1: /*
 2:      マウスに関するいろいろな設定を変更する
 3:
 4:      ダイアログとして呼び出して使用する
 5:
 6:      コンパイル（リンク）時に次のオプションが必要
 7:
 8:          XC  → /Y
 9:          GCC → -liocs
10:
11:                    (C) 中森 章, Jul.28, 1991
12: */
13: #include <stdio.h>
14: #define __POINT_T
15: #include <sxlib.h>
16: #define FALSE  0
17: #define TRUE   ~FALSE
18: #define MK_POINT(x,y)   (x<<16)+y
19:
20: drawMultiVal(v)
21: int v;
22: {
23:      rect r={184,84,184+5*6,84+18};
24:      point_t p={184+3,86};
25:      char BUF[10];
26:      sprintf(BUF,"%4.1f",(double)v/256.0);
27:      GMShadowRect(&r);
28:      GMMove(p);
29:      GMDrawStrZ(BUF);
30: }
31:
32: SetMSEnv()
33: {
34:      window  *dialogPtr;
35:      rect    winRect,ctrRect;
36:      event   eventRec;
37:      MsRec   *MsRecPtr;
38:
39:      control **selHdl;
40:      control **obsHdl;
41:      control **setHdl;
42:      control **canHdl;
43:
44:      control **rightHdl;
45:      control **horizHdl;
46:      control **vertiHdl;
47:      control **multiHdl;
48:
49:      char    rightval;
50:      char    horizval;
51:      char    vertival;
52:      short   multival;
```



▶やっとロードス島戦記が発売される。でも，手に入れるまでは安心できない（また延びたりして）。
鈴木　広志(23)栃木県

```
 53:
 54:     point_t pt;
 55:     int endDLOG;
 56:     int part;
 57:
 58:     winRect.left   = 260;
 59:     winRect.top    = 100;
 60:     winRect.right  = winRect.left+224;
 61:     winRect.bottom = winRect.top +132;
 62:
 63:     dialogPtr=WMOpen(NULL,&winRect,(LASCII*)"",TRUE,(38<<4
),
 64:                 (window *)-1,TRUE,TSGetID());
 65:     GMSetGraph(dialogPtr);
 66:
 67:     MsRecPtr=MSGetCurMsr(); /* マウスレコードへのポインタ */
 68:     rightval=B_BPEEK(&MsRecPtr->msRvsSwitch);
 69:     horizval=B_BPEEK(&MsRecPtr->msRvsLeftRight);
 70:     vertival=B_BPEEK(&MsRecPtr->msRvsForBack);
 71:     multival=MSMultiGet();
 72:
 73:     pt.x_y=MK_POINT(64,4);
 74:     GMMove(pt); GMDrawStrZ("マウスの環境設定");
 75:
 76:     ctrRect.left   = 8+24;
 77:     ctrRect.top    = 24;
 78:     ctrRect.right  = ctrRect.left+24+(6*014);
 79:     ctrRect.bottom = ctrRect.top +16;
 80:     rightHdl=CMOpen(dialogPtr,&ctrRect,(LASCII*)"¥14右利きマ
ウス",
 81:             TRUE,rightval&1,0,1,CI_OTNBTN<<4,0);
 82:
 83:     ctrRect.left   = 8+24;
 84:     ctrRect.top    = 44;
 85:     ctrRect.right  = ctrRect.left+24+(6*020);
 86:     ctrRect.bottom = ctrRect.top +16;
 87:     horizHdl=CMOpen(dialogPtr,&ctrRect,(LASCII*)"¥20左右移動
方向反転",
 88:             TRUE,horizval&1,0,1,CI_OTNBTN<<4,0);
 89:
 90:
 91:     ctrRect.left   = 8+24;
 92:     ctrRect.top    = 64;
 93:     ctrRect.right  = ctrRect.left+24+(6*020);
 94:     ctrRect.bottom = ctrRect.top +16;
 95:     vertiHdl=CMOpen(dialogPtr,&ctrRect,(LASCII*)"¥20上下移動
方向反転",
 96:             TRUE,vertival&1,0,1,CI_OTNBTN<<4,0);
 97:
 98:     ctrRect.left   = 8+(6*010)+4;
 99:     ctrRect.top    = 84;
100:     ctrRect.right  = ctrRect.left+120;
101:     ctrRect.bottom = ctrRect.top +16;
102:     multiHdl=CMOpen(dialogPtr,&ctrRect,(LASCII*)"",
103:             TRUE,multival,16,1024,CI_SLDVOL<<4,0);
104:
105:     pt.x_y=MK_POINT(8,84);
106:     GMMove(pt); GMDrawStrZ("移動係数");
107:     drawMultiVal(multival);
108:
109:     ctrRect.left   = 28;
110:     ctrRect.top    = 108;
111:     ctrRect.right  = ctrRect.left+(6*012);
112:     ctrRect.bottom = ctrRect.top +18;
113:     obsHdl=CMOpen(dialogPtr,&ctrRect,(LASCII*)"¥12 一時消去
",
114:             TRUE,0,0,1,CI_STDBTN<<4,0);
115:
116:     ctrRect.left   = 100;
117:     ctrRect.top    = 108;
118:     ctrRect.right  = ctrRect.left+(6*010);
119:     ctrRect.bottom = ctrRect.top +18;
120:     canHdl=CMOpen(dialogPtr,&ctrRect,(LASCII*)"¥10 取  消 "
121:             TRUE,0,0,1,CI_STDBTN<<4,0);
122:
123:     ctrRect.left   = 160;
124:     ctrRect.top    = 108;
125:     ctrRect.right  = ctrRect.left+(6*010);
126:     ctrRect.bottom = ctrRect.top +18;
127:     setHdl=CMOpen(dialogPtr,&ctrRect,(LASCII*)"¥10 設  定 "
,
128:             TRUE,0,0,1,CI_STDBTN<<4,0);
129:
130:     CMDraw(dialogPtr);
131:
132:     endDLOG=FALSE;
133:     while( !endDLOG ){
134:         EMGet(EM_EVERY,&eventRec);/* イベントを見る */
135:         switch( eventRec.eWhat ){
136:         case E_MSLDOWN:
137:             pt=eventRec.eWhere; /* マウスの座標 */
138:             if(pt.p.x<winRect.left || pt.p.x>winRect.right
139:             || pt.p.y<winRect.top  || pt.p.y>winRect.botto
m){
140:                 DMBeep(2);  /* ダイアログの外側 */
141:                 break;
142:             }
143:             part=SXCallCtrlM(dialogPtr,&eventRec,
144:                 NULL,NULL,NULL,&selHdl);
145:             if(selHdl==setHdl){      /* 設定 */
146:                 endDLOG=TRUE;
147:                 rightval=CMValueGet(rightHdl);
148:                 horizval=CMValueGet(horizHdl);
149:                 vertival=CMValueGet(vertiHdl);
150:                 multival=CMValueGet(multiHdl);
151:             }
152:             else if(selHdl==canHdl){     /* 取り消し */
153:                 endDLOG=TRUE;
154:             }
155:             else if(selHdl==multiHdl){
156:                 drawMultiVal(CMValueGet(multiHdl));
157:             }
158:             else if(selHdl==obsHdl){     /* 一時消去 */
159:                 MSObscureCsr();
160:             }
161:             break;
162:         case E_KEYDOWN:
163:             if((short)(eventRec.eWhom)!=13)
164:                 break;
165:             endDLOG=TRUE;             /* 設定に同じ */
166:             CMShine(setHdl,C_INBTTN);
167:             rightval=CMValueGet(rightHdl);
168:             horizval=CMValueGet(horizHdl);
169:             vertival=CMValueGet(vertiHdl);
170:             multival=CMValueGet(multiHdl);
171:             selHdl=setHdl;
172:             break;
173:         }
174:     }
175:     if(selHdl==setHdl){ /* 実際に値を設定 */
176:         B_BPOKE(&MsRecPtr->msRvsSwitch, -rightval);
177:         B_BPOKE(&MsRecPtr->msRvsLeftRight,  -horizval);
178:         B_BPOKE(&MsRecPtr->msRvsForBack,    -vertival);
179:         MSMultiSet(multival);
180:     }
181:
182:     MSInitCsr();      /* 右利き/左利きでパターンを反転させるため */
183:
184:     CMDispose(obsHdl);
185:     CMDispose(setHdl);
186:     CMDispose(canHdl);
187:     CMDispose(rightHdl);
188:     CMDispose(horizHdl);
189:     CMDispose(vertiHdl);
190:     CMDispose(multiHdl);
191:     WMDispose(dialogPtr);
192: }
```

## リスト3

```
  1: /*
  2:         【マウスカーソルのパターン】
  3:
  4:         このプログラムは CANDY 氏作成の SXmscsr.x の
  5:
  6:         マウスカーソルのパターンを一部参考にしています。
  7:
  8:         なお、
  9:
 10:         このファイルはXC(Ver2)でコンパイルしてください。
 11:
 12:         (GCCやXCのVer1ではコンパイルできません)
 13: */
 14: #define __POINT_T
 15: #include <sxlib.h>
 16:
 17: TXcsr cross_cs={ /* ×印 */
 18:     {7,7},
 19:     {
 20:      0b1110000000000111
 21:     ,0b1111000000001111
 22:     ,0b1111100000011111
 23:     ,0b0111110000111110
 24:     ,0b0011111001111100
 25:     ,0b0001111111111000
 26:     ,0b0000111111110000
 27:     ,0b0000011111100000
 28:     ,0b0000011111100000
 29:     ,0b0000111111110000
 30:     ,0b0001111111111000
 31:     ,0b0011111001111100
 32:     ,0b0111110000111110
 33:     ,0b1111100000011111
 34:     ,0b1111000000001111
 35:     ,0b1110000000000111
 36:     },
 37:     {
 38:      0b0000000000000000
 39:     ,0b0110000000000110
 40:     ,0b0111000000001110
 41:     ,0b0011100000011100
 42:     ,0b0001110000111000
 43:     ,0b0000111001110000
 44:     ,0b0000011111100000
 45:     ,0b0000001111000000
 46:     ,0b0000001111000000
 47:     ,0b0000011111100000
 48:     ,0b0000111001110000
 49:     ,0b0001110000111000
 50:     ,0b0011100000011100
 51:     ,0b0111000000001110
 52:     ,0b0110000000000110
 53:     ,0b0000000000000000
 54:
 55:     ,0b0000000000000000
 56:     ,0b0110000000000110
 57:     ,0b0111000000001110
 58:     ,0b0011100000011100
 59:     ,0b0001110000111000
 60:     ,0b0000111001110000
 61:     ,0b0000011111100000
 62:     ,0b0000001111000000
 63:     ,0b0000001111000000
 64:     ,0b0000011111100000
 65:     ,0b0000111001110000
 66:     ,0b0001110000111000
 67:     ,0b0011100000011100
 68:     ,0b0111000000001110
 69:     ,0b0110000000000110
 70:     ,0b0000000000000000
 71:
 72:     ,0b0000000000000000
 73:     ,0b0000000000000000
 74:     ,0b0000000000000000
 75:     ,0b0000000000000000
 76:     ,0b0000000000000000
 77:     ,0b0000000000000000
 78:     ,0b0000000000000000
 79:     ,0b0000000000000000
 80:     ,0b0000000000000000
 81:     ,0b0000000000000000
 82:     ,0b0000000000000000
 83:     ,0b0000000000000000
 84:     ,0b0000000000000000
 85:     ,0b0000000000000000
 86:     ,0b0000000000000000
 87:     ,0b0000000000000000
 88:
 89:     ,0b1110000000000111
 90:     ,0b1111000000001111
 91:     ,0b1111100000011111
 92:     ,0b0111110000111110
 93:     ,0b0011111001111100
 94:     ,0b0001111111111000
 95:     ,0b0000111111110000
 96:     ,0b0000011111100000
 97:     ,0b0000011111100000
 98:     ,0b0000111111110000
 99:     ,0b0001111111111000
100:     ,0b0011111001111100
101:     ,0b0111110000111110
102:     ,0b1111100000011111
103:     ,0b1111000000001111
104:     ,0b1110000000000111
105:     }
```

▶ 8月号の特集で浜崎氏が狂気の沙汰だ，といっていたことをやってしまいました。3日と10時間，ははは。

天達　雄一(16)京都府

```
106: };
107:
108: TXcsr peng_cs={ /* ペンギン */
109:       {1,6},
110:       {
111:        0b0000000111100000
112:       ,0b0000001111110000
113:       ,0b0000001111110000
114:       ,0b0000111111111000
115:       ,0b0000111111111000
116:       ,0b0011111111111100
117:       ,0b0111111111111100
118:       ,0b0001111111111100
119:       ,0b0000111111111100
120:       ,0b0000111111111100
121:       ,0b0000111111111110
122:       ,0b0000111111111111
123:       ,0b0000111111111111
124:       ,0b0001111111111111
125:       ,0b0011111111111101
126:       ,0b0000000000000000
127:       },
128:       {
129:        0b0000000111100000
130:       ,0b0000001111110000
131:       ,0b0000001001110000
132:       ,0b0000011000111000
133:       ,0b0000010000111000
134:       ,0b0011101000011100
135:       ,0b0111000011001100
136:       ,0b0001100111100100
137:       ,0b0000100111110100
138:       ,0b0000100111111100
139:       ,0b0000100111111110
140:       ,0b0000100011111111
141:       ,0b0000100000111111
142:       ,0b0001110011000011
143:       ,0b0011111111111101
144:       ,0b0000000000000000
145:
146:       ,0b0000000111100000
147:       ,0b0000001111110000
148:       ,0b0000001001110000
149:       ,0b0000011000111000
150:       ,0b0000010000111000
151:       ,0b0011101000011100
152:       ,0b0111000011001100
153:       ,0b0001100111100100
154:       ,0b0000100111110100
155:       ,0b0000100111111100
156:       ,0b0000100111111110
157:       ,0b0000100011111111
158:       ,0b0000100000111111
159:       ,0b0001110011000011
160:       ,0b0011111111111101
161:       ,0b0000000000000000
162:
163:       ,0b0000000000000000
164:       ,0b0000000000000000
165:       ,0b0000000000000000
166:       ,0b0000000000000000
167:       ,0b0000000000000000
168:       ,0b0000000000000000
169:       ,0b0000000000000000
170:       ,0b0000000000000000
171:       ,0b0000000000000000
172:       ,0b0000000000000000
173:       ,0b0000000000000000
174:       ,0b0000000000000000
175:       ,0b0000000000000000
176:       ,0b0000000000000000
177:       ,0b0000000000000000
178:       ,0b0000000000000000
179:
180:       ,0b0000000111100000
181:       ,0b0000001111110000
182:       ,0b0000001111110000
183:       ,0b0000111111111000
184:       ,0b0000111111111000
185:       ,0b0011111111111100
186:       ,0b0111111111111100
187:       ,0b0001111111111100
188:       ,0b0000111111111100
189:       ,0b0000111111111100
190:       ,0b0000111111111110
191:       ,0b0000111111111111
192:       ,0b0000111111111111
193:       ,0b0001111111111111
194:       ,0b0011111111111101
195:       ,0b0000000000000000
196:       }
197: };
198:
199: TXcsr vsh_cs={ /* VS. X */
200:       {1,1},
201:       {
202:        0b1100000000000000
203:       ,0b1110000000000000
204:       ,0b1111000000000000
205:       ,0b1111100000000000
206:       ,0b1111110000000000
207:       ,0b1111111000000000
208:       ,0b1111111100000000
209:       ,0b1111111110000000
210:       ,0b1111111111000000
211:       ,0b1111111111100000
212:       ,0b1111111000000000
213:       ,0b1110111100000000
214:       ,0b1100111100000000
215:       ,0b0000011110000000
216:       ,0b0000011110000000
217:       ,0b0000001100000000
218:       },
219:       {
220:        0b0000000000000000
221:       ,0b0100000000000000
222:       ,0b0110000000000000
223:       ,0b0111000000000000
224:       ,0b0111100000000000
225:       ,0b0111110000000000
226:       ,0b0111111000000000
227:       ,0b0111111100000000
228:       ,0b0111111110000000
229:       ,0b0111110000000000
230:       ,0b0110110000000000
231:       ,0b0100011000000000
232:       ,0b0000011000000000
233:       ,0b0000001100000000
234:       ,0b0000001100000000
235:       ,0b0000000000000000
236:
237:       ,0b0000000000000000
238:       ,0b0100000000000000
239:       ,0b0110000000000000
240:       ,0b0111000000000000
241:       ,0b0111100000000000
242:       ,0b0111110000000000
243:       ,0b0111111000000000
244:       ,0b0111111100000000
245:       ,0b0111111110000000
246:       ,0b0111110000000000
247:       ,0b0110110000000000
248:       ,0b0100011000000000
249:       ,0b0000011000000000
250:       ,0b0000001100000000
251:       ,0b0000001100000000
252:       ,0b0000000000000000
253:
254:       ,0b0000000000000000
255:       ,0b0000000000000000
256:       ,0b0000000000000000
257:       ,0b0000000000000000
258:       ,0b0000000000000000
259:       ,0b0000000000000000
260:       ,0b0000000000000000
261:       ,0b0000000000000000
262:       ,0b0000000000000000
263:       ,0b0000000000000000
264:       ,0b0000000000000000
265:       ,0b0000000000000000
266:       ,0b0000000000000000
267:       ,0b0000000000000000
268:       ,0b0000000000000000
269:       ,0b0000000000000000
270:
271:       ,0b1100000000000000
272:       ,0b1110000000000000
273:       ,0b1111000000000000
274:       ,0b1111100000000000
275:       ,0b1111110000000000
276:       ,0b1111111000000000
277:       ,0b1111111100000000
278:       ,0b1111111110000000
279:       ,0b1111111111000000
280:       ,0b1111111111100000
281:       ,0b1111111000000000
282:       ,0b1110111100000000
283:       ,0b1100111100000000
284:       ,0b0000011110000000
285:       ,0b0000011110000000
286:       ,0b0000001100000000
287:       }
288: };
289:
290: TXcsr hand_cs={ /* 手形 */
291:       {1,5},
292:       {
293:        0b0000000000000000
294:       ,0b0000000000000000
295:       ,0b0000000000000000
296:       ,0b0000001111111100
297:       ,0b0111111111111110
298:       ,0b1111111111111111
299:       ,0b0111111111111111
300:       ,0b0001111111111111
301:       ,0b0000111111111111
302:       ,0b0000111111111111
303:       ,0b0000011111111111
304:       ,0b0000001111111111
305:       ,0b0000000111111110
306:       ,0b0000000000000000
307:       ,0b0000000000000000
308:       ,0b0000000000000000
309:       },
310:       {
311:        0b0000000000000000
312:       ,0b0000000000000000
313:       ,0b0000000000000000
314:       ,0b0000001111111000
315:       ,0b0111111000001110
316:       ,0b1000001100000011
317:       ,0b0111111111100001
318:       ,0b0001000001010001
319:       ,0b0000111111001101
320:       ,0b0000100001000001
321:       ,0b0000011111000001
322:       ,0b0000001001000011
323:       ,0b0000000111111110
324:       ,0b0000000000000000
325:       ,0b0000000000000000
326:       ,0b0000000000000000
327:
328:       ,0b0000000000000000
329:       ,0b0000000000000000
330:       ,0b0000000000000000
331:       ,0b0000001111111000
332:       ,0b0111111000001110
333:       ,0b1000001100000011
334:       ,0b0111111111100001
335:       ,0b0001000001010001
336:       ,0b0000111111001101
337:       ,0b0000100001000001
338:       ,0b0000011111000001
339:       ,0b0000001001000011
340:       ,0b0000000111111110
341:       ,0b0000000000000000
342:       ,0b0000000000000000
343:       ,0b0000000000000000
344:
345:       ,0b0000000000000000
346:       ,0b0000000000000000
347:       ,0b0000000000000000
348:       ,0b0000000000000000
349:       ,0b0000000000000000
350:       ,0b0000000000000000
351:       ,0b0000000000000000
352:       ,0b0000000000000000
353:       ,0b0000000000000000
354:       ,0b0000000000000000
355:       ,0b0000000000000000
356:       ,0b0000000000000000
357:       ,0b0000000000000000
358:       ,0b0000000000000000
359:       ,0b0000000000000000
360:       ,0b0000000000000000
361:
362:       ,0b0000000000000000
363:       ,0b0000000000000000
364:       ,0b0000000000000000
365:       ,0b0000001111111100
366:       ,0b0111111111111110
367:       ,0b1111111111111111
368:       ,0b0111111111111111
369:       ,0b0001111111111111
370:       ,0b0000111111111111
371:       ,0b0000111111111111
372:       ,0b0000011111111111
373:       ,0b0000001111111111
374:       ,0b0000000111111110
375:       ,0b0000000000000000
376:       ,0b0000000000000000
377:       ,0b0000000000000000
378:       }
379:
380: };
381: TXcsr mouse_cs={ /* ネズミ */
382:       {2,7},
383:       {
384:        0b0000000000000000
385:       ,0b0000000000000000
386:       ,0b0000000000000000
387:       ,0b0000111100000000
388:       ,0b0001111111000100
389:       ,0b0011111111100010
390:       ,0b0111111111111001
391:       ,0b1111111111111001
392:       ,0b1111111111111001
393:       ,0b1111111111110011
394:       ,0b1111111111111110
395:       ,0b0000000000000000
396:       ,0b0000000000000000
397:       ,0b0000000000000000
398:       ,0b0000000000000000
399:       ,0b0000000000000000
400:       },
401:       {
402:        0b0000000000000000
403:       ,0b0000000000000000
404:       ,0b0000000000000000
405:       ,0b0000111100000000
406:       ,0b0001000111000100
407:       ,0b0011111000100010
408:       ,0b0100000000011001
409:       ,0b1000000000001001
410:       ,0b1010000000001001
411:       ,0b1000000000010011
412:       ,0b1111111111111110
413:       ,0b0000000000000000
414:       ,0b0000000000000000
415:       ,0b0000000000000000
416:       ,0b0000000000000000
417:       ,0b0000000000000000
418:
419:       ,0b0000000000000000
420:       ,0b0000000000000000
421:       ,0b0000000000000000
422:       ,0b0000111100000000
423:       ,0b0001000111000100
424:       ,0b0011110000100010
425:       ,0b0100000000011001
426:       ,0b1011000000001001
427:       ,0b1011000000001001
428:       ,0b1000000000010011
429:       ,0b1111111111111110
430:       ,0b0000000000000000
431:       ,0b0000000000000000
432:       ,0b0000000000000000
433:       ,0b0000000000000000
434:       ,0b0000000000000000
435:
436:       ,0b0000000000000000
437:       ,0b0000000000000000
438:       ,0b0000000000000000
439:       ,0b0000000000000000
440:       ,0b0000000000000000
441:       ,0b0000000000000000
442:       ,0b0000000000000000
443:       ,0b0000000000000000
444:       ,0b0000000000000000
445:       ,0b0000000000000000
446:       ,0b0000000000000000
447:       ,0b0000000000000000
448:       ,0b0000000000000000
449:       ,0b0000000000000000
450:       ,0b0000000000000000
451:       ,0b0000000000000000
452:
453:       ,0b0000000000000000
454:       ,0b0000000000000000
455:       ,0b0000000000000000
456:       ,0b0000111100000000
457:       ,0b0001111111000100
458:       ,0b0011111111100010
459:       ,0b0111111111111001
```

▶ 付録ディスクもいいけど，本当は入力したほうが楽しい（ときもある）。確かに手間は省けて楽だけど，その分知識が身につかない（人もいる）ということを知っていますか？

太田　哲雄(16)北海道

```
460:        ,0b1111111111111001
461:        ,0b1111111111111001
462:        ,0b1111111111110011
463:        ,0b1111111111111110
464:        ,0b0000000000000000
465:        ,0b0000000000000000
466:        ,0b0000000000000000
467:        ,0b0000000000000000
468:        ,0b0000000000000000
469:        }
470: };
471:
472: TXcsr hudson_cs={ /* ハドソン */
473:        {3,3},
474:        {
475:         0b0100011001000100
476:        ,0b0010111010001000
477:        ,0b0011111101010000
478:        ,0b0111111111100000
479:        ,0b0111111111100000
480:        ,0b1111111111111000
481:        ,0b0111111111111100
482:        ,0b0011111111111110
483:        ,0b0101111111111110
484:        ,0b1001111111111110
485:        ,0b0010111111111111
486:        ,0b0100111111111110
487:        ,0b1000011111111110
488:        ,0b0000001111111100
489:        ,0b0000000111110000
490:        ,0b0000000000000000
491:        },
492:        {
493:         0b0100011001000100
494:        ,0b0010101010001000
495:        ,0b0011001101010000
496:        ,0b0101110111100000
497:        ,0b0100101100100000
498:        ,0b1111011100011000
499:        ,0b0101111000111100
500:        ,0b0010100000111110
501:        ,0b0101000001110010
502:        ,0b1001100011110010
503:        ,0b0010111111100011
504:        ,0b0100111111000010
505:        ,0b1000011000000110
506:        ,0b0000001100011100
507:        ,0b0000000111110000
508:        ,0b0000000000000000
509:
510:        ,0b0100011001000100
511:        ,0b0010101010001000
512:        ,0b0011001101010000
513:        ,0b0101110111100000
514:        ,0b0100101100100000
515:        ,0b1111011100011000
516:        ,0b0101111000111100
517:        ,0b0010100000111110
518:        ,0b0101000001110010
519:        ,0b1001100011110010
520:        ,0b0010111111100011
521:        ,0b0100111111000010
522:        ,0b1000011000000110
523:        ,0b0000001100011100
524:        ,0b0000000111110000
525:        ,0b0000000000000000
526:
527:        ,0b0000000000000000
528:        ,0b0000000000000000
529:        ,0b0000000000000000
530:        ,0b0000000000000000
531:        ,0b0000000000000000
532:        ,0b0000000000000000
533:        ,0b0000000000000000
534:        ,0b0000000000000000
535:        ,0b0000000000000000
536:        ,0b0000000000000000
537:        ,0b0000000000000000
538:        ,0b0000000000000000
539:        ,0b0000000000000000
540:        ,0b0000000000000000
541:        ,0b0000000000000000
542:        ,0b0000000000000000
543:
544:        ,0b0100011001000100
545:        ,0b0010111010001000
546:        ,0b0011111101010000
547:        ,0b0111111111100000
548:        ,0b0111111111100000
549:        ,0b1111111111111000
550:        ,0b0111111111111100
551:        ,0b0011111111111110
552:        ,0b0101111111111110
553:        ,0b1001111111111110
554:        ,0b0010111111111111
555:        ,0b0100111111111110
556:        ,0b1000011111111110
557:        ,0b0000001111111100
558:        ,0b0000000111110000
559:        ,0b0000000000000000
560:
561:        }
562:
563: };
564:
565: TXcsr shadow_cs={ /* 影付き */
566:        {1,1},
567:        {
568:         0b1100000000000000
569:        ,0b1110000000000000
570:        ,0b1111000000000000
571:        ,0b1111100000000000
572:        ,0b1111110000000000
573:        ,0b1111111000000000
574:        ,0b1111111100000000
575:        ,0b1111111110000000
576:        ,0b1111111111000000
577:        ,0b1111111111100000
578:        ,0b1111111111110000
579:        ,0b1111111111111000
580:        ,0b1111111111111100
581:        ,0b1111111000000000
582:        ,0b1111000000000000
583:        ,0b1100000000000000
584:        },
585:        {
586:         0b1100000000000000
587:        ,0b1010000000000000
588:        ,0b1101000000000000
589:        ,0b1100100000000000
590:        ,0b1110010000000000
591:        ,0b1110001000000000
592:        ,0b1111000100000000
593:        ,0b1111100010000000
594:        ,0b1111100001000000
595:        ,0b1111110000100000
596:        ,0b1111100000010000
597:        ,0b1111000000001000
596:        ,0b1111100000010000
597:        ,0b1111000000001000
598:        ,0b1111000111111100
599:        ,0b1110110000000000
600:        ,0b1101000000000000
601:        ,0b1100000000000000
602:
603:        ,0b1100000000000000
604:        ,0b1010000000000000
605:        ,0b1001000000000000
606:        ,0b1000100000000000
607:        ,0b1000010000000000
608:        ,0b1000001000000000
609:        ,0b1000000100000000
610:        ,0b1000000010000000
611:        ,0b1000000001000000
612:        ,0b1000000000100000
613:        ,0b1000011000010000
614:        ,0b1000111111111000
615:        ,0b1000111111111100
616:        ,0b1001111000000000
617:        ,0b1011000000000000
618:        ,0b1100000000000000
619:
620:        ,0b0000000000000000
621:        ,0b0000000000000000
622:        ,0b0000000000000000
623:        ,0b0000000000000000
624:        ,0b0000000000000000
625:        ,0b0000000000000000
626:        ,0b0000000000000000
627:        ,0b0000000000000000
628:        ,0b0000000000000000
629:        ,0b0000000000000000
630:        ,0b0000000000000000
631:        ,0b0000000000000000
632:        ,0b0000000000000000
633:        ,0b0000000000000000
634:        ,0b0000000000000000
635:        ,0b0000000000000000
636:
637:        ,0b1100000000000000
638:        ,0b1110000000000000
639:        ,0b1111000000000000
640:        ,0b1111100000000000
641:        ,0b1111110000000000
642:        ,0b1111111000000000
643:        ,0b1111111100000000
644:        ,0b1111111110000000
645:        ,0b1111111111000000
646:        ,0b1111111111100000
647:        ,0b1111111111110000
648:        ,0b1111111111111000
649:        ,0b1111111111111100
650:        ,0b1111111000000000
651:        ,0b1111000000000000
652:        ,0b1100000000000000
653:        }
654: };
```

## リスト4

```
  1: /*
  2:        【マウスカーソルのパターン（アニメーション用）】
  3:
  4:        このプログラムは CANDY 氏作成の SXmscsr.x の
  5:
  6:        マウスカーソルのパターンを流用しています。
  7:
  8:        なお、
  9:
 10:        このファイルはXC（Ver2）でコンパイルしてください。
 11:
 12:        （GCCやXCのVer1ではコンパイルできません）
 13: */
 14: #define __POINT_T
 15: #include <sxlib.h>
 16:
 17: static TXcsr apat1={
 18:        {1,1},
 19:        {
 20:         0b1100000000000000
 21:        ,0b1110000000000000
 22:        ,0b1111000000000000
 23:        ,0b1111100000000000
 24:        ,0b1111110000000000
 25:        ,0b1111111000000000
 26:        ,0b1111111100000000
 27:        ,0b1111111110000000
 28:        ,0b1111111111000000
 29:        ,0b1111111111100000
 30:        ,0b1111111000000000
 31:        ,0b1110111100000000
 32:        ,0b1100111100000000
 33:        ,0b0000011110000000
 34:        ,0b0000011110000000
 35:        ,0b0000001100000000
 36:        },
 37:        {
 38:         0b0000000000000000
 39:        ,0b0000000000000000
 40:        ,0b0000000000000000
 41:        ,0b0000000000000000
 42:        ,0b0000000000000000
 43:        ,0b0000000000000000
 44:        ,0b0000000000000000
 45:        ,0b0111111100000000
 46:        ,0b0111111110000000
 47:        ,0b0111110000000000
 48:        ,0b0110110000000000
 49:        ,0b0100011000000000
 50:        ,0b0000011000000000
 51:        ,0b0000001100000000
 52:        ,0b0000001100000000
 53:        ,0b0000000000000000
 54:
 55:        ,0b0000000000000000
 56:        ,0b0100000000000000
 57:        ,0b0110000000000000
 58:        ,0b0111000000000000
 59:        ,0b0000000000000000
 60:        ,0b0000000000000000
 61:        ,0b0000000000000000
 62:        ,0b0000000000000000
 63:        ,0b0000000000000000
 64:        ,0b0000000000000000
 65:        ,0b0110110000000000
 66:        ,0b0100011000000000
 67:        ,0b0000011000000000
 68:        ,0b0000001100000000
 69:        ,0b0000001100000000
 70:        ,0b0000000000000000
 71:
 72:        ,0b0000000000000000
 73:        ,0b0100000000000000
 74:        ,0b0110000000000000
 75:        ,0b0111000000000000
 76:        ,0b0111100000000000
 77:        ,0b0111110000000000
 78:        ,0b0111111000000000
 79:        ,0b0111111100000000
 80:        ,0b0111111110000000
 81:        ,0b0111110000000000
 82:        ,0b0110110000000000
 83:        ,0b0100011000000000
 84:        ,0b0000011000000000
 85:        ,0b0000000000000000
 86:        ,0b0000000000000000
 87:        ,0b0000000000000000
 88:
 89:        ,0b1100000000000000
 90:        ,0b1110000000000000
 91:        ,0b1111000000000000
 92:        ,0b1111100000000000
 93:        ,0b1111110000000000
 94:        ,0b1111111000000000
 95:        ,0b1111111100000000
 96:        ,0b1111111110000000
 97:        ,0b1111111111000000
 98:        ,0b1111111111100000
 99:        ,0b1111111000000000
100:        ,0b1110111100000000
101:        ,0b1100111100000000
102:        ,0b0000011110000000
103:        ,0b0000011110000000
104:        ,0b0000001100000000
105:        }
106: };
107:
108: static TXcsr apat2={
109:        {1,1},
110:        {
111:         0b1100000000000000
112:        ,0b1110000000000000
113:        ,0b1111000000000000
114:        ,0b1111100000000000
115:        ,0b1111110000000000
116:        ,0b1111111000000000
117:        ,0b1111111100000000
118:        ,0b1111111110000000
119:        ,0b1111111111000000
120:        ,0b1111111111100000
121:        ,0b1111111000000000
122:        ,0b1110111100000000
123:        ,0b1100111100000000
124:        ,0b0000011110000000
125:        ,0b0000011110000000
126:        ,0b0000001100000000
127:        },
128:        {
129:         0b0000000000000000
130:        ,0b0100000000000000
131:        ,0b0110000000000000
132:        ,0b0111000000000000
133:        ,0b0000000000000000
134:        ,0b0000000000000000
135:        ,0b0000000000000000
```

▶ 徹夜のデバッグで病気になっちゃいそうです。いまほど自分の職業を呪ったことはない。
早崎　拓也(19)千葉県

```
136:        ,0b0000000000000000
137:        ,0b0000000000000000
138:        ,0b0000000000000000
139:        ,0b0110110000000000
140:        ,0b0100011000000000
141:        ,0b0000011000000000
142:        ,0b0000001100000000
143:        ,0b0000001100000000
144:        ,0b0000000000000000
145:
146:        ,0b0000000000000000
147:        ,0b0000000000000000
148:        ,0b0000000000000000
149:        ,0b0000000000000000
150:        ,0b0111100000000000
151:        ,0b0111110000000000
152:        ,0b0111111000000000
153:        ,0b0000000000000000
154:        ,0b0000000000000000
155:        ,0b0000000000000000
156:        ,0b0000000000000000
157:        ,0b0000000000000000
158:        ,0b0000000000000000
159:        ,0b0000001100000000
160:        ,0b0000001100000000
161:        ,0b0000000000000000
162:
163:        ,0b0000000000000000
164:        ,0b0000000000000000
165:        ,0b0000000000000000
166:        ,0b0000000000000000
167:        ,0b0111100000000000
168:        ,0b0111110000000000
169:        ,0b0111111000000000
170:        ,0b0111111100000000
171:        ,0b0111111110000000
172:        ,0b0111110000000000
173:        ,0b0110110000000000
174:        ,0b0100011000000000
175:        ,0b0000011000000000
176:        ,0b0000001100000000
177:        ,0b0000001100000000
178:        ,0b0000000000000000
179:
180:        ,0b1100000000000000
181:        ,0b1110000000000000
182:        ,0b1111000000000000
183:        ,0b1111100000000000
184:        ,0b1111110000000000
185:        ,0b1111111000000000
186:        ,0b1111111100000000
187:        ,0b1111111110000000
188:        ,0b1111111111000000
189:        ,0b1111111111100000
190:        ,0b1111111000000000
191:        ,0b1110111100000000
192:        ,0b1100111100000000
193:        ,0b0000011110000000
194:        ,0b0000011110000000
195:        ,0b0000001100000000
196:        }
197: };
198:
199: static TXcsr apat3={
200:        {1,1},
201:        {
202:         0b1100000000000000
203:        ,0b1110000000000000
204:        ,0b1111000000000000
205:        ,0b1111100000000000
206:        ,0b1111110000000000
207:        ,0b1111111000000000
208:        ,0b1111111100000000
209:        ,0b1111111110000000
210:        ,0b1111111111000000
211:        ,0b1111111111100000
212:        ,0b1111111000000000
213:        ,0b1110111100000000
214:        ,0b1100111100000000
215:        ,0b0000011110000000
216:        ,0b0000011110000000
217:        ,0b0000001100000000
218:        }
219:        {
220:         0b0000000000000000
221:        ,0b0100000000000000
222:        ,0b0110000000000000
223:        ,0b0111000000000000
224:        ,0b0111100000000000
225:        ,0b0111110000000000
226:        ,0b0111111000000000
227:        ,0b0000000000000000
228:        ,0b0000000000000000
229:        ,0b0000000000000000
230:        ,0b0000000000000000
231:        ,0b0000000000000000
232:        ,0b0000000000000000
233:        ,0b0000001100000000
234:        ,0b0000001100000000
235:        ,0b0000000000000000
236:
237:        ,0b0000000000000000
238:        ,0b0100000000000000
239:        ,0b0110000000000000
240:        ,0b0111000000000000
241:        ,0b0111100000000000
242:        ,0b0111110000000000
243:        ,0b0111111100000000
244:        ,0b0111111110000000
245:        ,0b0111111111000000
246:        ,0b0111110000000000
247:        ,0b0000000000000000
248:        ,0b0000000000000000
249:        ,0b0000000000000000
250:        ,0b0000000000000000
251:        ,0b0000000000000000
252:        ,0b0000000000000000
253:
254:        ,0b0000000000000000
255:        ,0b0100000000000000
256:        ,0b0110000000000000
257:        ,0b0111000000000000
258:        ,0b0000000000000000
259:        ,0b0000000000000000
260:        ,0b0000000000000000
261:        ,0b0111111110000000
262:        ,0b0111111111000000
263:        ,0b0111110000000000
264:        ,0b0000000000000000
265:        ,0b0000000000000000
266:        ,0b0000000000000000
267:        ,0b0000000000000000
268:        ,0b0000000000000000
269:        ,0b0000000000000000
270:
271:        ,0b1100000000000000
272:        ,0b1110000000000000
273:        ,0b1111000000000000
274:        ,0b1111100000000000
275:        ,0b1111110000000000
276:        ,0b1111111000000000
277:        ,0b1111111100000000
278:        ,0b1111111110000000
279:        ,0b1111111111000000
280:        ,0b1111111111100000
281:        ,0b1111111000000000
282:        ,0b1110111100000000
283:        ,0b1100111100000000
284:        ,0b0000011110000000
285:        ,0b0000011110000000
286:        ,0b0000001100000000
287:        }
288: };
289:
290: static TXcsr apat4={
291:        {1,1},
292:        {
293:         0b1100000000000000
294:        ,0b1110000000000000
295:        ,0b1111000000000000
296:        ,0b1111100000000000
297:        ,0b1111110000000000
298:        ,0b1111111000000000
299:        ,0b1111111100000000
300:        ,0b1111111110000000
301:        ,0b1111111111000000
302:        ,0b1111111111100000
303:        ,0b1111111000000000
304:        ,0b1110111100000000
305:        ,0b1100111100000000
306:        ,0b0000011110000000
307:        ,0b0000011110000000
308:        ,0b0000001100000000
309:        },
310:        {
311:         0b0000000000000000
312:        ,0b0100000000000000
313:        ,0b0110000000000000
314:        ,0b0111000000000000
315:        ,0b0111100000000000
316:        ,0b0111110000000000
317:        ,0b0111111000000000
318:        ,0b0111111100000000
319:        ,0b0111111110000000
320:        ,0b0111110000000000
321:        ,0b0000000000000000
322:        ,0b0000000000000000
323:        ,0b0000000000000000
324:        ,0b0000000000000000
325:        ,0b0000000000000000
326:        ,0b0000000000000000
327:
328:        ,0b0000000000000000
329:        ,0b0000000000000000
330:        ,0b0000000000000000
331:        ,0b0000000000000000
332:        ,0b0111100000000000
333:        ,0b0111110000000000
334:        ,0b0111111000000000
335:        ,0b0111111100000000
336:        ,0b0111111110000000
337:        ,0b0111110000000000
338:        ,0b0110110000000000
339:        ,0b0100011000000000
340:        ,0b0000011000000000
341:        ,0b0000000000000000
342:        ,0b0000000000000000
343:        ,0b0000000000000000
344:
345:        ,0b0000000000000000
346:        ,0b0100000000000000
347:        ,0b0110000000000000
348:        ,0b0111000000000000
349:        ,0b0111100000000000
350:        ,0b0111110000000000
351:        ,0b0111111000000000
352:        ,0b0000000000000000
353:        ,0b0000000000000000
354:        ,0b0000000000000000
355:        ,0b0110110000000000
356:        ,0b0100011000000000
357:        ,0b0000011000000000
358:        ,0b0000001100000000
359:        ,0b0000001100000000
360:        ,0b0000000000000000
361:
362:        ,0b1100000000000000
363:        ,0b1110000000000000
364:        ,0b1111000000000000
365:        ,0b1111100000000000
366:        ,0b1111110000000000
367:        ,0b1111111000000000
368:        ,0b1111111100000000
369:        ,0b1111111110000000
370:        ,0b1111111111000000
371:        ,0b1111111111100000
372:        ,0b1111111000000000
373:        ,0b1110111100000000
374:        ,0b1100111100000000
375:        ,0b0000011110000000
376:        ,0b0000011110000000
377:        ,0b0000001100000000
378:        }
379: };
380:
381: static TXcsr apat5={
382:        {1,1},
383:        {
384:         0b1100000000000000
385:        ,0b1110000000000000
386:        ,0b1111000000000000
387:        ,0b1111100000000000
388:        ,0b1111110000000000
389:        ,0b1111111000000000
390:        ,0b1111111100000000
391:        ,0b1111111110000000
392:        ,0b1111111111000000
393:        ,0b1111111111100000
394:        ,0b1111111000000000
395:        ,0b1110111100000000
396:        ,0b1100111100000000
397:        ,0b0000011110000000
398:        ,0b0000011110000000
399:        ,0b0000001100000000
400:        },
401:        {
402:         0b0000000000000000
403:        ,0b0000000000000000
404:        ,0b0000000000000000
405:        ,0b0000000000000000
406:        ,0b0111100000000000
407:        ,0b0111110000000000
408:        ,0b0111111000000000
409:        ,0b0111111100000000
410:        ,0b0111111110000000
411:        ,0b0111110000000000
412:        ,0b0110110000000000
413:        ,0b0100011000000000
414:        ,0b0000011000000000
415:        ,0b0000000000000000
416:        ,0b0000000000000000
417:        ,0b0000000000000000
418:
419:        ,0b0000000000000000
420:        ,0b0000000000000000
421:        ,0b0000000000000000
422:        ,0b0000000000000000
423:        ,0b0000000000000000
424:        ,0b0000000000000000
425:        ,0b0000000000000000
426:        ,0b0111111100000000
427:        ,0b0111111110000000
428:        ,0b0111110000000000
429:        ,0b0110110000000000
430:        ,0b0100011000000000
431:        ,0b0000011000000000
432:        ,0b0000001100000000
433:        ,0b0000001100000000
434:        ,0b0000000000000000
435:
436:        ,0b0000000000000000
437:        ,0b0100000000000000
438:        ,0b0110000000000000
439:        ,0b0111000000000000
440:        ,0b0111100000000000
441:        ,0b0111110000000000
442:        ,0b0111111000000000
443:        ,0b0111111100000000
444:        ,0b0111111110000000
445:        ,0b0111110000000000
446:        ,0b0000000000000000
447:        ,0b0000000000000000
448:        ,0b0000000000000000
449:        ,0b0000001100000000
450:        ,0b0000001100000000
451:        ,0b0000000000000000
452:
453:        ,0b1100000000000000
454:        ,0b1110000000000000
455:        ,0b1111000000000000
456:        ,0b1111100000000000
457:        ,0b1111110000000000
458:        ,0b1111111000000000
459:        ,0b1111111100000000
460:        ,0b1111111110000000
461:        ,0b1111111111000000
462:        ,0b1111111111100000
463:        ,0b1111111000000000
464:        ,0b1110111100000000
465:        ,0b1100111100000000
466:        ,0b0000011110000000
467:        ,0b0000011110000000
468:        ,0b0000001100000000
469:        }
470: };
471:
472: static TXcsr *apat_p1=&apat1,
473:        *apat_p2=&apat2,
474:        *apat_p3=&apat3,
475:        *apat_p4=&apat4,
476:        *apat_p5=&apat5;
477:
478: TXcsr **msapat[]={
479:        &apat_p1,
480:        &apat_p2,
481:        &apat_p3,
482:        &apat_p4,
483:        &apat_p5
484: };
```

▶Yetのハイスコアですが，現在64,050点が最高です。　清水　昌一(23)愛知県

特集

# マシン語との邂逅

コンピュータに何かをやらせようというときには，当然コンピュータに実行させたい内容を伝えなければならない。人と人とのコミュニケーションに言語が使われるように，コンピュータの世界でも言語と呼ばれる記号の羅列が用いられる。
コンピュータ言語にもいろいろあるが，コンピュータはすべての言語を直接理解してくれるわけではない。高級言語と呼ばれるBASICやC言語などは，実行するときに翻訳しながら，あるいはいったん翻訳してしまったものを実行する。
では，何に翻訳されるのか。いわずもがな，マシン語である。マシン語，機械語という言葉に聞き覚えのない人はいないと思う。
BASICやC言語をコンパイルしてもマシン語のプログラムは得られるが，そこにはまだ無駄がある。無駄のない，高速なプログラムを書きたいときには直接マシン語を書く。あるいはコンパイルしたあとで最適化するということが必要である。
コンピュータと付き合っていくうちにマシン語と巡りあう。もちろん，ずっと巡りあわずに進んでいく人もいる。しかし，マシン語との邂逅は，よりいっそう充実した付き合いをもたらしてくれるものと期待したい。

CONTENTS

基本用語解説
# マシン語の考え方

Nakano Shuichi 中野 修一

速い！　小さい！　マシン語は究極の言語です。それだけにマシン語を扱うには幅広い知識も要求されてきます。まずはコンピュータがどうやって動いているのか見当もつかない，という人のための基本用語解説です。

## マシン語を使うということ

X68000は“68000”というCPU名を冠したパソコンです。普段から知らず知らずのうちにCPUが68000であることの恩恵を受けてはいるのですが（というより86系でないことの恩恵），大半のユーザーにはCPUの違いを感じることはほとんどないといっていいでしょう。

X68000の特徴は，

68000CPU
比較的広いメモリ
強力なI/O
純国産OS

に絞られます。メモリとOSについては普段からお世話になっていますし，I/Oまわりは市販ゲームが証明しています。CPUについては，アセンブラが使えるプログラマがユーザーの約15％というあたりに特徴が表れているといえるでしょうか。

C言語全盛の世間ではアセンブラを使うことはどんどん少なくなっています。しかし，強力なI/Oの真価を見るためにはマシン語は欠かせません。マシン語の最大の魅力はなんといっても速度です。10MHzのクロックというのは1秒間に10,000,000回のクロックを刻みます。

68000は4から170クロックで1命令を実行しますから，1命令平均20クロックとしても秒間500,000個の命令が実行できます。ちなみにアクションゲームなどでは1/30秒単位くらいで一連の処理を行います。20クロックというのは遅いようにも思われますが，このあいだに「インデックスつきプログラムカウンタ相対アドレッシングで示されるメモリ上の32ビットデータを任意のデータレジスタに加算する」という処理が可能です。

68000の魅力は豊富なレジスタと充実したアドレッシングモードです。これらは最適化の弱いコンパイラではほとんど使用されません。その真価はアセンブラで開発を行うことによって初めて引き出されるといえます。

マシン語は難解だという評価についてはいまさらなにもいいません。一方で「なんでもできるマシン語」というイメージを持ちながらも，マイクロプロセッサでできることというのは本当に限られています。すなわち，

データの移動
四則演算
論理演算
シフト/ローテート
比較
ジャンプ
サブルーチンコール
⋮

などです。これらをコツコツと組み合わせて，多彩な処理を実現していくことがすなわちマシン語プログラミングです。

## move：データの移動

実際のアセンブリ言語プログラムを見るとほとんどがデータの移動（move命令）で占められていることがわかります。

move命令は，

move A，B

のように使用されます。これはBASICの，

B＝A

と同じ意味を持ちます。「AをBに移す」わけですが，A，Bの部分にはさまざまなものが指定されます。

通常，CPUがアクセスするプログラムやデータは1バイトずつ16進数6桁で示されるアドレスに整然と格納されています。マシン語ではアドレス（住所）を指示するのに，単に○丁目○○番地というのではなく，「突き当たりのタバコ屋から3軒目のお向かい」とかいった指定も可能なのです。このデータの場所の指定方法のことをアドレッシングモードといいます。これについては後ろの記事で解説しています。

とりあえず，マシン語ではデータを移動することが非常に多くなります。あらかじめ決められている手順で，要求されているパラメータをセットしていくことが処理の大半を占めることも少なくありません。

たとえば，IOCSコールなら，指定されたレジスタに適当なデータを入れ，TRAP命令を実行します。DOSコールならスタックに適当なデータを揃えて決められたFラインの未実装命令というものを実行することになるでしょう（SXコールならAラインを使う）。I/Oを直接制御する場合でも周辺チップのレジスタに適当なデータをセットして実行開始のコマンドを送るなどの操作をします。どれも基本的には「要求されたデータをセットして実行」というパターンになっています。

ここで「TRAP命令」や「Fライン未実装命令」などについて頭を悩ませる必要はありません。BASICでラインを引くときにラインのアルゴリズムを知らなくても大丈夫なように，これらの命令の細かい動作についてなにも知らなくても，書式どおり記述すれば必要な機能はちゃんと得ることができます。

## 演算

さて，データの移動がマシン語プログラムの大半を占めることはわかりました。ここで問題になるのは先ほど「適当なデータ」を，と誤魔化していた部分です。データをこの「適当なデータ」に変換する部分を作ることこそが真にプログラミングといえる部分でしょう。この部分ではお決まりの命令によるおまじないではなく，プログラマ各位のアルゴリズムとコーディングの技術が問われてきます。

算術演算と論理演算などはBASICやC言語でもお馴染みですね。算術演算はともかく論理演算については使わずにすます人も

いるかもしれません。しかし，マシン語の場合，ビット単位の操作が多くなりますので，論理演算は必須科目です。

## cmp，b［CC］：比較，分岐

情報の最小量をビットといいます。1ビットは「Yes.」または「No.」ひとつで答える情報です。「今期のセリーグ優勝チームはどこか？」という問いに1ビットで答えることはできませんが，「今期の優勝チームはカープに決まったか？」という問いには1ビットで十分です。

コンピュータ言語で記述できる条件文は，二者択一が基本です。これがまさに，1と0に対応した動作をするのがマシン語での条件分岐なのです。それも，

「○○と××は同じか？」

「レジスタは0になったか？」

など，1ビットで答えられるような「特定の質問」に問題を分解していくことがマシン語プログラミングの骨組みにほかなりません。

cmp（compare）で比較してb［CC］（branch Condition Code：CCには条件が入る）で分岐するというのがひとつのパターンです。コンピュータ言語では必ずお目にかかる制御構造（IF～THENやREPEAT～UNTILなど）もこれらの組み合わせで作成します。Cコンパイラをお持ちの方は，コンパイルされたあとのアセンブラソースで制御構造がどのように実現されているのか見てみるのもよいでしょう。

## レジスタというもの

68000CPUは「内部32ビット」の16ビットCPUだといわれています。これはどういうことなのでしょうか？

CPUはメモリ上にあるプログラムやデータを読み込んで処理するということは皆さんご存じでしょう。大雑把にいってレジスタとは，メモリから読み込んだデータを処理するための場所のことです。演算のほとんどはレジスタで行われます。

図1は68000のレジスタを示すものです。アドレスレジスタとデータレジスタに分かれていますね。レジスタ1個には最大32ビット幅のデータを格納できます。

32ビットデータというのは，

&B10010000011010110010010001111101

のようなものです。CPUにデータを与えるということは，この1と0の塊でゲジゲジ（CPUの足）のデータバスという線のそれぞれを一斉にON/OFFしていることになります。

レジスタ内では，8ビット，16ビット，32ビットのデータそれぞれをほとんど同等に扱えます。これがソフトウェア的に32ビットといわれるゆえんです。しかし，ハードウェア的には，一度に扱えるデータは16ビットに制限されています。実は32ビットデータは2回に分けて処理されているのです（使っている分にはわからないが）。

＊　＊　＊

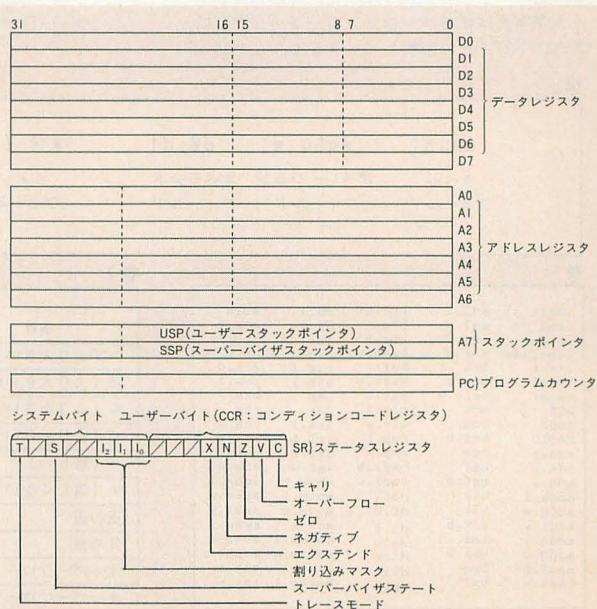
レジスタ幅32ビットでは0～4,294,967,295の数値を扱うことができます。32ビットというのがかなりの幅を持っていることはわかりましたが，いくら大きくても有限の値ではなんでもできるというわけにはいきません。これで不都合は起きないのでしょうか？

たとえば，32ビット目一杯の数字，

&HFFFFFFFF

を2つ足してみましょう。BASICならたちまちエラーが発生するところですが，マシン語の場合はいちいちエラーで停止するようなことはありません。ではどうなるかというと，あふれでた部分を無視してレジスタに演算結果が残ります。

10進数で考えてみましょう。ある4桁の数字が2つあったとして，これらを足すともしかしたら5桁の数字になってしまうかもしれません。しかし，5桁目は“1”以外の数値になることはありえないのです。ですから，桁あふれを起こしたかどうかさえわかれば，演算の結果は正確にわかります。68000ではCCRレジスタのキャリビットで桁あふれを知ることができます（CCRはZ80のFレジスタに相当）。引き算で負の数値になった場合も同様です。

掛け算の場合はどうでしょうか？　幸か不幸か，68000では16ビット×16ビットを32ビットで扱う掛け算しかサポートされていません。これは完全に32ビット幅のレジスタで収まります。

桁あふれ以外にも，演算結果がレジスタの範囲で収まらなかった場合には同様にCCRの各フラグに状況が反映されます。ただし，データが32ビットを超えた幅を持っている場合には処理プログラムを作成しなければなりません。

ソフトウェア的に32ビットであるということは32ビット以下のデータに対しては非常に効率よく作業ができるということを意味しています。

## サービスルーチンの使用

DOSコール，IOCSコール，SXコール，FLOATやOPMDRVなどの拡張コール……X68000にはメーカーによって用意されたたくさんのサービスルーチンがあります。IOCSなどを使っていたのではマシン語を使う意味が半減すると考える人もいますが，とりあえず使えるものは使ってみるのが正解でしょう。

IOCS.X以前の文字表示やグラフィックルーチンなどはあまり使いものにならなかったものの，基本的なルーチンは劇的に改善されるとは考えにくいといえます。先月掲載されたMAGICでも常駐処理などにDOSコール，DMA制御などにIOCSコールを使っています。

＊　＊　＊

68000は10年前のアーキテクチャによるCPUです。内部処理などは最新のものに比べて見劣りする部分もあります。美しいといわれた基本構成もV60/70などと比べれば不徹底な感じもするでしょう。にもかかわらず，10年たったいまでも現役で使用されているというのはどうしてでしょうか。

2年でひと昔という業界の常識では，パフォーマンスで追いつくはずがないのに，

図1

結構互角にわたりあっています。基本的なアーキテクチャの素性がよかった，といってしまえばそれまでですが，10年前に作った仕様がCISCで最高のパフォーマンスを示す68040でも同じまま保たれているというのは驚異的です。

自由度の大きい素直な構成，高い信頼性，プログラムの組みやすさ……実にエンスー好みの石です。実際，これほど多くの人から愛されたCPUはちょっとないでしょう。68000CPUは使い込むに値する石だといえます。

マシン語を使うことは難しいと思い込んでいる人もいます。しかし，なんらかの言語でプログラムが組めればマシン語への移行はさほど困難なことではないでしょう。余計な気負いをなくせば，マシン語は単純で素直な横顔を見せてくれるはずです。

## アセンブラソースリストを読む

いきなりですが，ここでアセンブラの基本，ソースリストををどのように読み下していくか，説明していくことにします。

初めて目にするソースリストは，英数字の羅列にしか見えません。といっても，コンピュータ言語は英語を元に命令が書かれている場合がほとんどですので，一応，発音することができるときもあるでしょう。ところが，読むことさえできないこともあります。それは，きちんとした単語を命令に使用していない，意味合いがコンピュータ世界の言葉で，一般的な意味合いと違っている，意味は同じでも具体的に何をしているかわからない場合などがそうです。コンピュータという独自の世界の中でのみ通用する言葉といえるかもしれません。

結構，自己流の読み方でもなんとかなるものですが，正しい読み方があるならばそれに沿うのが賢い選択でしょう。変なところで恥をかかずにすみますしね。

**●ソースリストの構成は？**

まず，表1を見てください。これはMAGICのソースリストで使われている命令をひと通り抜き出したものです。いわゆるオペレーションフィールドに書かれているやつですね。

と，ここで妙な単語が出てきました。オペレーションフィールドとはなんぞや。これは図1を参照してください。ソースリストを記述するためには，いくつかの決まりがあって，1行は4つのフィールドで書かれることになります。

これらは何カラム(桁)目から書き始めなさい，と厳密に指定されているものではありません。区切りは文字列ごとのスペース/タブコードで行われます。アセンブラは，行の先頭から文字列が始まっていたらその文字列をラベルと解釈し，次のスペースを見つけたら，その次に出てくる文字列はオペレーションコード……という具合に1行ごとに解釈していきます。これはソースリストを書くための基本フォーマットですから，まず知っておかなくてはならないことです。

**●オペコードの読み方**

で，問題となるのがオペレーションフィールドに書かれているニーモニックの読み方です。基本は英単語の省略されたものなので，かなりハナモゲラな英字が並んでいます。

省略の仕方はそれぞれの単語の頭文字を取ったり，母音を省略したり，なかにはそのままのもあります。

ここで，少し例を挙げてみます。

- LEA……Load Effective Address
- MOVEM……Move Multiple registers
- MULS……Multiply as Signed
- SUBI……Subtract Immediate

とまあ，こんな感じです。こういうふうに書かれると，それぞれどういった命令だか見当がつきやすくなるでしょう。実際に読むときには，この省略前の単語をすべて発音するのが正しいはずです。でも，日本人である我々にはちょっと面倒臭いですね。通常の会話ではほとんどの場合，無理やりローマ字読みするか，その機能を直接言葉でいうことになります。

ただ，先ほどいったように，元の意味を知っていればその命令を理解するのにも役立つし，他人に説明するときにも説明しやすくなるでしょうから，覚えておきましょう。

**●オマケの意味**

次にニーモニックをさらに細かく見ていくことにします。まず目につくのはニーモニックの最後についているデータサイズです。これはニーモニックが処理するデータサイズを指定するものです。全部で4つのサイズ，

.l……ロングワード（4バイト）
.w……ワード（2バイト）
.b……バイト（1バイト）
.s……ショート（1/2バイト）

を指定することができます。たとえば，

MOVE.W　　d0,d1

の場合には，d0レジスタからd1レジスタへワードサイズの転送を行う，というようないい方をします。ちなみに最後のショートサイズのデータは，Bcc,BSR命令のみに使用します。

データサイズは必ず指定する必要がなく，オペランドのサイズが固定されている場合には省略することもできます。また，サイズ指定が必要であってもワードサイズのアクセスをする場合には省略が可能です。とはいっても，変に省略するクセがついてしまい，肝心なところでつけ忘れるようなことがあっては本末転倒です。なるべくつけるようにしましょう。

もうひとつ，表1のMOVE，ADDなどを見て気づくと思いますが，“a”とか“i”とかついているでしょう。これらは，

a……Address
i……Immediate
q……Quick
m……Memory，Multiple

の頭文字です。要するに，どのオペランドの種類に影響を与えるか明確にするためのものです。これらは明確にしなくても，アセンブラが勝手に解釈してくれるので意味さえ把握しておけばいいでしょう。

**●コンディションコード**

最後にコンディションコードの説明をします。これは各命令で影響されるフラグによって，制御するために必要なものです。おもに条件分岐命令で使われ，表2のようにまとめられます。プログラムがどのように制御されているかを知るためには，このコンディションコードを覚えておかなくてはなりません。

直前の命令でキャリフラグ，ゼロフラグがどのように変化するか判断して，それぞれの状況に応じて分岐先へと制御を移していくか確かめながらソースを眺めていきます。条件が成立したときと，そうでないときの両方を把握していかないと，どこで分岐して流れが変わったのか理解できなくなってしまいます。結構，慣れないときついものがあるので意地でもコンディションコードをしっかり覚えましょう。

＊　　＊　　＊

ということで，ニーモニックを中心に，アセンブラソースリストがどのように構成されているか説明してきました。あとは疑似命令を理解すれば，とりあえずソースリストがわけのわからない文字の羅列である，という概念から脱出できることと思います。

しかし，命令が理解できたからといってプログラムを理解することにはなりません。プログラムを読み下すということは，そのプログラムを理解するということですから。そして，他人のプログラムを読み下せるようになれば，自分のプログラミングテクニックもかなり向上するでしょう。がんばってください。　（M. H.）

図1

| main: | subq.w | d0,d1 | *メインルーチン |
|---|---|---|---|
| ラベル<br>フィールド | オペレーション<br>フィールド | オペランド<br>フィールド | コメント<br>フィールド |

表1

```
.data      asl.l    bpl.b    ds.l      muls
.end       asl.w    bra      ds.w      neg.w
.even      asr.l    bra.b    eori.b    nop
.include   asr.w    bset.b   exg       or.b
.text      bcc      bsr      exg.l     ori.w
.xdef      bcc.b    btst.b   ext.l     pea.l
.xref      bclr.b   clr.b    jmp       rte
DOS        bcs      clr.l    jsr       rts
IOCS       beq      clr.w    lea.l     scs
MAGIC      beq.b    cmp.l    lsl.l     scs.b
add.b      bge      cmp.w    lsl.w     sub.l
add.l      bgt      cmpi.b   lsr.w     sub.w
add.w      bgt.b    cmpi.w   move.b    suba.l
adda.l     ble      dbf      move.l    suba.w
adda.w     blt      dc.b     move.w    subi.w
addi.w     blt.b    dc.l     movea.l   subq.l
addq.l     bmi      dc.w     movem.l   subq.w
addq.w     bmi.b    divs     movem.w   swap
andi.b     bne      divu     moveq     tst.b
andi.w     bpl      ds.b     moveq.l   tst.w
```

表2

| 条件 | cc | |
|---|---|---|
| | 符号あり | 符号なし |
| >（より大きい） | HI | GT |
| ≧（より大きいか等しい） | CC | GE |
| ≦（より小さいか等しい） | LS | LE |
| <（より小さい） | CS | LT |
| ＝（等しい） | EQ | |
| ≠（等しくない） | NE | |
| 正の値 | PL | －－ |
| 負の値 | MI | －－ |
| オーバーフロー | VS | |
| オーバーフローでない | VC | |

X-BASICからアセンブラへ、という方々に贈る

# IOCS, DOSコールをうまく使おう

Mounai Toshiyuki　毛内 俊行

X-BASICでバリバリとプログラムを組んでいるものの，やっぱりスピードが気になる。そういう場合には「XBAStoC」が有用だが，自分でマシン語に置き換えていく，という作業もいいだろう。このとき役に立つのがIOCS，DOSコールだ。

## あなたがアセンブラを使う理由

いきなり質問。なぜ君はアセンブラを使おうとするのか。アセンブラを使わなくても，X68000にはX-BASICという便利な言語がある。X-BASICならばC言語ライクな記述でプログラムが書けるし，音楽からスプライトまでX68000の機能を隅々まで使うことができる。そう，唯一の欠点といえるスピードも，コンパイラを使えばほとんど解決できるのだ。それでもスピードに不満のあるユーザーなら，まず，C compiler PRO-68K（以下，XCと略）を買うだろう。

ところが，こういう動機でXCを買ったのだから，当然Cの勉強なんてやらない。つまり，BASICコンパイラとしてしか使わないのだ。

定価44,000円（税別）もの大金を払って買うのだから，最初からそんなもったいない使い方をしようと思って買う人はいないだろう。XCを購入したときは，誰もが志を高くかかげ，Cのエキスパートになる日を夢見たはずである。夜空に輝く星に向かって「歴史に残る傑作を作るぞ！」と誓った人だっているかもしれない。

ところが，その志も気がついたら流れ星と一緒にどこか遠いところへと飛んでいってしまいがち。「のどもと過ぎれば熱さを忘れる」の例にあるように，大金を払ったことなどすっかり忘れているのだ。

まあ，本人はいつか必ずCの勉強をしようと思っているかもしれない。ではなぜやらないのだろうか。理由は簡単。XCについてくる，あの分厚いマニュアルを読むのが面倒だからである。やがてマニュアルは部屋の隅でほこりをかぶり，XCはただのBASICコンパイラとなってしまうのだった。

そんな日々を送っている自分に，ひょんなことから変化が訪れる。いつも使っているXCのシステムディスクの中に，アセンブラとリンカを見つけるのである。

「おお，これこそ俺が探し求めていたものだ。ようし，今日からアセンブラをやるぞっ！」

こうして，決意を新たにアセンブラの勉強を始めるのだ。なんのことはない。アセンブラなら，あのXCのライブラリマニュアルをただの１ページだって読む必要がないだけなのだ。つまりただの逃避である。

X-BASICユーザーがアセンブラをやる理由には，このように自然と楽なほうへ逃げた結果ということもあるかもしれない。少なくとも私にはその傾向は十分にある。

しかし，世の中そんなに甘くはない。いつまでも川の流れに身をまかせていたら，滝から落ちて大怪我をする。アセンブラは大きな滝つぼ。ここから這い上がるのは，決して甘くはない。そこでこれから，滝つぼの底から這い上がるためのアセンブラ講座が始まるのだ。

## 準備をする

まず，アセンブラを勉強するために必要なものを揃えなければいけない。とりあえず必要なファイルを紹介しよう。

### ●AS.X

いわずと知れたアセンブラである。これがないと何もできない。シャープ純正パッケージの中では，XCを購入するのがいちばん手っ取り早いだろう。以前なら，THE福袋V2.0というアセンブラによる開発ソフトが売られいたが，XCのバージョンアップにともなって店頭から消えたようだ。そのほかにも巷では，AS.Xの高速版HAS.Xが出回っているので，それを入手するのもひとつの手であろう。

### ●LK.X

リンカである。AS.Xとともに，これがなくては話にならない。アセンブラで作成した複数のオブジェクトファイルをリンクして，ひとつの実行ファイルを作る（もっともオブジェクトファイルがひとつしかなくても，これを使わないと実行ファイルは作成できない）。これも，XCを買うとついている。高速版のHLK.Xも通信などで簡単に入手できるだろう。

### ●ED.X

ご存じスクリーンエディタである。プログラムを書くために必要であるが，特に用意しなくてもX68000のシステムディスクに入っている。知らなかった人はあわててシステムディスクを覗いて見ることだ。もちろん，このほかにmicroEMACSを持っている人はそれを使うこともできるし，その気になればWP.Xでも大丈夫だ。

### ●IOCSCALL.MAC，DOSCALL.MAC

この２つのファイルはX 68000やHuman 68kの内部ルーチンを呼び出すために使う。なくてもプログラムは書けるが，あるととても便利だ。使い方はあとで説明するので，とりあえず用意してほしい。このファイルもXCのシステムディスクに入っている。また，本誌の付録ディスクにも入っていたので，持っていない人はこちらから入手してほしい。

今回はこれだけ用意すれば十分である。このほかに，アセンブラを使う人に便利なファイルとしてはDB.Xという便利なデバッガがあるが，こちらは今回の特集で泉氏が詳しく説明しているのでそちらを参照してほしい。

必要なファイルを用意したら，次は必要な書物を用意しよう。いわゆる参考書というやつだ。これは最低２冊は必要である。

１冊目はMPU68000の解説書である。つまり，68000でマシン語を使うために必要なことが書いてある本のことだ。大きな書店に行けば，必ず数種類の本を見つけることができるので，自分の読みやすい本を探そう。選ぶ決め手は，68000の命令セットの説明があることだ（当たり前）。

ちなみに私は「68000プログラマーズハン

ドブック」(技術評論社) を使っている。

また,XCのアセンブラマニュアルも命令セットが載っているので(アルファベット順であまり実用的ではないが),すでにXCを持っていて,金銭的に新しい本を買うのが困難な人はこれを使うといい。

ただし,同じXCでも,ver.1.0のマニュアルにかぎる。ver.2.0のマニュアルはデバッガの解説など,いろんな説明が増えた関係で,命令セットは小さな表になって巻末に追いやられている。

2冊目はX68000の解説書だ。68000の仕様がわかっても,X68000の機種に依存した部分がわからなければプログラムはまるで書けない。つまり,DOSコールやIOCSコールがまとめられていて,アクセスの方法やプログラムのノウハウを解説した本が必要である。いま私がいちばん愛用しているのは,XCについてくる「プログラマーズマニュアル」だ。内部コールの説明はもちろん,X-BASICの外部関数からデバイスドライバの作り方まで丁寧に説明してある。

これ以外でよくまとめられている本を挙げると,「X68000環境ハンドブック」(工学社) がある。IOCSコールやDOSコールなど,一部独自解析したものまで載っており,XCを持っていないユーザーは一見の価値があると思う。とりあえず,書店で実際に見て気に入った本を探すこと。これがいちばん大切なことなのだ。本を揃えたら,準備は終わりだ。

## ウォーミングアップ

準備は終わったが,まだまだプログラミングには入れない。噺家だって最初は雑巾がけから始まるのだ。ここでは,アセンブラを使うために必要な知識として,レジスタとスタックについて軽く触れておこう。

### ●レジスタ

レジスタとは「MPUが数字を記憶しておく箱」というのが,いちばん簡単な説明だろう。68000でふだん我々が使うレジスタはd0~d7のデータレジスタと呼ばれる8本のレジスタと,a0~a7のアドレスレジスタ8本の計16本だ。アドレスレジスタとデータレジスタの違いは,読んで字の如く,レジスタに入る数がメモリ上のアドレスを示すものか,任意のデータかの違いだけだ。

こうやって見るとレジスタは,BASICでいうところの変数と使い方がよく似ている。しかし,レジスタと変数は別のものだと考えたほうがいい。たしかに,小さいプログラムを作るときなら,レジスタを変数のように使うのが常套手段であるが,先ほどもいったようにレジスタは全部で16本しかないのだ。変数16個のプログラムがどんなものか考えれば,レジスタだけでは十分ではないということがわかるだろう。

それでは,レジスタは実際にどんな使われ方をしているのかというと,MPU68000では(特にX68000では)主に,IOCSコールとのデータのやりとりを行うために使われている。詳しい使い方は,あとでどんどん出てくるので,とりあえずレジスタがどんなものなのかということを覚えておいてほしい。

### ●スタック

「♪変数のようで変数でない。レジスタのようでレジスタでない。それは何かと尋ねたら,あ,スタック,スタック」

そう,レジスタでも変数でもない,BASICでは見たことがない形式のデータ格納庫がスタックと呼ばれるものだ。スタックのデータはメモリ上に1列に並んでいる。そして,その先頭のアドレスがスタックポインタと呼ばれるレジスタ(68000ではa7レジスタ)に格納されている。

面白いのはスタックにデータを格納すると,新しく格納されたデータはスタックのいちばん前に置かれるということだ。

たとえば,書類を入れる段ボール箱を想像してみよう。最初,段ボール箱に書類が入っていないとすると,箱の中は空である。そこに書類を1枚入れてみると,箱の底に書類が1枚入ることになる。2枚目を入れると,新しく入れた書類は1枚目の書類の上に置かれる。1枚目の書類を取るときには,まず,上に乗っている2枚目の書類を取り除いてからでないと取ることができない。同様に100枚の書類を入れた箱から最初の1枚目の書類を取るには,上に乗っている99枚の書類を取らなくてはいけない。このような構造をLIFO (Last in first out)という。スタックはこのLIFO構造のデータで構成されており,最初に入れたデータはいちばん最後に,そしていちばん最後に入れたデータを最初に取り出すことができるのである。

Z80のアセンブラを使っていた人は,スタックがCALL命令のリターンアドレスなども格納していて,スタックを扱うときには入れたデータの数と取り出したデータの数を管理しておかなければ暴走する恐れがあることを知っていると思う。

68000ではユーザースタックとシステムスタックの2本のスタックを持っていて,サブルーチンコールのリターンアドレスなどはシステムスタックに,我々がデータを格納するのは(一般に)ユーザースタックへと区別されているので,特に暴走の心配をすることはないが,スタックの管理をおろそかにすると思わぬバグが発生することになる。

また,X68000ではHuman68kのDOSコールのデータ受け渡しに,このスタックが使われている。

## プログラミングの実際

いよいよ本番である。どんなプログラムを入力してもいいのだが,やはり実行したときに結果がすぐに見られるように,画面に何か文字を表示するプログラムがいいだろう。

まず,リスト1を見てみよう。見慣れたX-BASICのプログラムだ。実行すると画面に“Hello boy!”と表示して終了する。わずか2行の簡単なプログラムだが,これをアセンブラで書いてみると,リスト2のようになる。初めて見る人にはなんだかまったくわからないだろうが,とりあえず入力してみよう。まず,キーボードから,

ED LIST2.S

と入力してエディタを立ち上げ,リスト2を入力しよう。拡張子のSは,アセンブラのソースファイルという意味だ。入力が終わったら,ディスクにセーブして,エディタから抜ける。

そして,

AS LIST2

を実行する。いま書いたプログラムをアセンブラでアセンブルするのだ。短いプログラムなので,作業はすぐに終了する。

エラーが出なければ,“LIST2.O”というファイルがあるはずだ。ディレクトリを見てみよう。これがいわゆるオブジェクトファイルというやつだ。

そうすると次は,リンカでこのオブジェクトファイルから実行ファイルを作らなく

リスト1

```
10 print"Hello boy!"
20 end
```

リスト2

```
1:      lea.l   msgadr,a1
2:      move.l  #$21,d0
3:      trap    #15
4:      .dc.w   $ff00
5: msgadr:
6:      .dc.b   'Hello boy!',$0d,$0a,0
```

てはいけない。

LK LIST2

を実行しよう。このようにしてようやく，実行ファイル“LIST2.X”が完成するのである。キーボードから，

LIST2

と実行してみよう。間違いがなければ，画面に“Hello boy！”と表示されるはずである。

さて，リスト2のようなプログラムを書けば，画面に文字を表示できることはわかった。しかし，リスト2にはまだちょっと謎が多い。そこで，リスト3を見てみよう。リスト3はリスト2とまったく同じ働きをするプログラムである。リスト2と違うところは，1行目と2行目に，

_B_PRINT： equ $21

_EXIT： equ $FF00

が加わったこと。そして，

move.l #$21,d0

と，

.dc.w $FF00

が，それぞれ，

move.l #_B_PRINT,d0

と，

.dc.w _EXIT

に変わっている点である。

察しのいい人なら，このリスト3とリスト2を見比べただけで，このプログラムの仕組みの半分は理解できたのではないかと思う。

キーワードは_B_PRINTと_EXITだ。なかでも，_B_PRINTはBASICのprint文とよく似ていて，文字表示の命令と怪しい関係にありそうである。結論からいうと，怪しいだけではなく本当に関係がある。_B PRINTはX68000のIOCSコールを使った，文字列表示ルーチンなのである。

IOCSコールとは，X68000のROM内にあらかじめプログラムされているサブルーチンの総称で，文字表示のほかにもキーボードやジョイスティックの入力，AD PCMやスプライトなんかも扱うことができる。ふだん，X-BASICで関数を使ってやっていたことなら，このIOCSコールを使うことでほぼ書き換えができるのだ。

さて，このIOCSコールの使い方だが，だいたい次のような手順で行われる。

1) 必要なデータをレジスタにセットする
2) d0レジスタにIOCSコール番号をセットする
3) trap #15を実行する

trap命令とは一種のサブルーチンコールだと思えばいい。リスト2では，

lea.l msgadr,a1

move.l #$21,d0

trap #15

の3行がIOCSコールのためのプログラムである。1行目と2行目でレジスタに必要な数値を格納して，trap命令を実行しているのがわかるだろう。

XCのマニュアルのB_PRINTのページを見ればわかるが，「a1レジスタに表示する文字列の先頭アドレスをセットし，d0レジスタに$21をセットしてtrap #15を実行すること」と書いてある。

d0にセットする$21という値は，B_PRINTのIOCSコール番号であるので，データとしてセットするのは文字列の先頭アドレスだけである。

気づいた人もいると思うが，プログラマーズマニュアルにはB_PRINTと書いてあり，_B_PRINTと書いてない。この1文字目の“_”は，プログラムを書くときに内部ルーチンのラベルか，それ以外のラベルかの区別をつけるために必要なものだと思ってくれればいい（少なくとも，私はそう思っている）。

さて，IOCSコールの話はこれくらいにして，今度はDOSコールの話である。DOSコールとは，X68000のOS，Human68kの中にあらかじめプログラムされた内部ルーチンである。OSに内蔵されているのだから，当然RAMにロードされる。IOCSコールがROMにプログラムされているのと，異なるところである。

リスト3を見てみよう。6行目に，

.dc.w _EXIT

とあるはずだ。これがDOSコールの呼び出しである。プログラマーズマニュアルを見てみると，exitはプログラムを終了するコールだと書いてある。リスト2を見てみると，この行は，

.dc.w $FF00

と同じであることがわかるから，Human68kではメモリ上に$FF00の数を見つけたら，プログラムを終了するようになっているのだ。マニュアルにはやはり“_”をつけず，しかも小文字で書いてあるが，プログラムを書くときは大文字で，“_”をつけて書いてほしい。なぜなら，このあとで説明するdoscall.macとiocscall.macを使ったプログラムを書くときのコール名の表記方法が，こうなっているからだ。

データの受け渡し方法だが，IOCSコールではレジスタだったのに対し，DOSコールではスタックを使う。ただし，このexitではOS側にデータを渡す必要がないので，ここでは使っていない。

さて，リスト4を見てみよう。プログラムが大きく変わり，わかりやすくなった。しかし，これはリスト2，リスト3とまったく同じプログラムである。ヒントは1行目と2行目にある“.include”という命令にある。これはインクルード命令といって，この命令のあとに指定したファイルを，読み込んで一緒にアセンブルするという命令である。このプログラムでは，iocscall.macというファイルと，doscall.macというファイルを，そこに読み込んでアセンブルしているのだ。

この2つのファイルは，カレントのディレクトリに置いておくのが第1の条件だが，as.xのver.2.0からは環境変数INCLUDEにこの2つのファイルを置いたディレクトリを指定しておくことによって，いつでもインクルードファイルを使うことができるようになった。ver.1.0を使っている人は，早く2.0を手に入れよう。

さて，4行目以降を見てみると，先ほどもいったようにプログラムがシンプルで見やすい。特にIOCSコールが，

lea.l msgadr,a1

IOCS _B_PRINT

の2行ですんでいるのはうれしい。これが，iocscall.macというファイルの力強いところだ。仕組みを知りたかったら，iocscall.macの中を覗いてみるといい。すると，リス

**リスト3**

```
1: _B_PRINT:      equ    $21
2: _EXIT:      equ    $FF00
3:      lea.l   msgadr,a1
4:      move.l  #_B_PRINT,d0
5:      trap    #15
6:      .dc.w   _EXIT
7: msgadr:
8:      .dc.b   'Hello boy!',$0d,$0a,0
```

**リスト4**

```
1: .include        iocscall.mac
2: .include        doscall.mac
3:
4:         lea.l   msgadr,a1
5:         IOCS    _B_PRINT
6:         DOS     _EXIT
7: msgadr:
8:         .dc.b   'Hello boy!',$0d,$0a,0
```

ト5に書かれたような箇所が見える。マクロ指定である。つまり,

IOCS なんちゃら

と書けば,"なんちゃら"というIOCSコールを実行するようになっているのだ。

同じように,doscall.macにもDOSコールのマクロが定義されているので,

DOS うんちゃら

と書けば"うんちゃら"というDOSコールを実行するのだ。

さて,そこでリスト6を見てほしい。実行すればわかるのだが,このプログラムも画面に文字を表示するプログラムである。いままでと違うところは,IOCSコールをいっさい使わずに書いてあるという点である。DOSコールしか使っていないのだ。その証拠に,iocscall.macはインクルード指定していない。

そして,_B_PRINTのかわりにDOSコールの_PRINTが使われている。プログラマーズマニュアルを見ると,printは文字列を表示すると書いてある。IOCSコールのB_PRINTとまったく同じ機能だ。ただし,先ほどもいったように,DOSコールではデータの受け渡しにスタックを使うので,IOCSコールとは少々異なった感じになる。

リスト6を見ると,

```
pea     msgadr
DOS     _PRINT
addq.l #4,sp
```

とあるが,これがデータの受け渡しをともなうDOSコールの使い方である。

最初,pea命令で文字列の先頭アドレスをスタックに積み,2行目でDOSコールを実行するので,一応それだけでDOSコールは文字列を表示してくれる。

しかし,先ほどスタックの説明でもいったとおり,スタックはデータをどんどん積み上げる構造なので,積み上げたデータは降ろさなくてはいけない。スタックにデータを積んだままでは,いらないデータがスタックに山のように溜まり,挙句の果てにメモリがなくなってしまうのである。

そこで最後に,"addq.l #4,sp"を実行しなくてはいけないのだ。これはスタックを管理するspレジスタ(a7レジスタ)に4を足す命令である。文字列のアドレスは4バイトなので,スタックに格納されたときにはspレジスタの値が4だけ小さくなる。そこで,使い終わったときにspレジスタに4を加えておけば,spの値はDOSコールを使う前の値に戻るというからくりだ。

DOSコールでは,データの大きさは4バイトと決まっているわけではないから,6バイトを積んだら6を,8バイト積んだら8を,使い終わったときに忘れずに足すように心がけなくてはいけない。

こんなところでDOSコールとIOCSコールの役割がわかったのではないかと思う。さて,実際にこのDOSコールとIOCSコールを使ってX-BASICのプログラムをアセンブラで書き換えた場合に,どのようになるのかを見てみよう。

まず,リスト7を見てほしい。リスト7はグラフィックを使ったサンプルで,画面にラインで美しい模様を描くプログラムである。内容は簡単で,ラインをぐるっと1周回転させるだけである。

リスト8がアセンブラで書いた同じプログラムである。理解しやすいように,X-BASICで書いたときの命令を,リストの右側にコメントの形で載せてある。X-BASICの命令とほとんど1対1で対応しているので,読みやすいと思う。ただ,肝心のラインを引くときに,

IOCS_LINE

となっていて,始点や終点などのパラメータがないのに疑問を感じる人もいるだろう。ユーザーズマニュアルを見るとわかるのだが,IOCSコールのLINEは座標などのパラメータを,a1レジスタの示すアドレスから12バイトのデータで持っているので,そのアドレスをあらかじめセットしておくだけでいいのである。

リスト8では,プログラム開始直後に,

lea.l linebf,a1

で指定している。唯一,X-BASICの命令で表現できないところだ(X-BASICにもポインタがあればいいのに)。とりあえず,X-BASICの関数やステートメントが,IOCSコールやDOSコールできれいに置き換えることができることを確認してほしい。lineはLINE,screenはCRTMODとG_CLR_ONで,endはexitになっているはずだ。

ただ,repeat~untilはアセンブラでは本来は書き表せない。リスト8も実際にはif~gotoという表記が正しいのだが,X-BASICではgotoが禁じ手になっているのであえてこう書いた。このように,X-BASICのだいたいの命令はIOCSコールとDOSコールによって置き換えることが可能である。

ただ,さっきB_PRINTをDOSコールのprintで置き換えたように,IOCSコールをDOSコールで置き換えることは,いつもできるとはかぎらない。

これはDOSコールとIOCSコールの役割の違いを考えればわかることなのだが,DOSコールはあくまでもHuman68kというOSのために作られているので,グラフィック表示や音楽演奏のような機種に依存した命令はサポートしていないのだ。つまり,リスト8に出てきたようなLINE命令はDOSコールには存在しない。

逆に,IOCSコールはファイルのアクセスなど,OSに依存した命令はサポートしていないし,プロセスを終了するexit命令だってDOSコールだからできる技なのだ。そのあたりを考えて2つを使い分けるのがいいだろう。

さて,最後にもうちょっと面白いプログラムを作ってみよう。IOCSコールの一覧を

**リスト6**

```
1: .include          doscall.mac
2:          pea      msgadr
3:          DOS      _PRINT
4:          addq.l   #4,sp
5:          DOS      _EXIT
6: msgadr:
7:          .dc.b    'Hello boy!',$0d,$0a,0
```

**リスト5**

```
1: IOCS    macro    callname
2:         move.l   #callname,d0
3:         trap     #15
4:         endm
```

**リスト7**

```
 10 int sx,sy,ex,ey,cl,ls
 20 /*
 30 screen 2,0,1,1
 40 /*
 50 sx = 0
 60 sy = 0
 70 ex = 767
 80 ey = 511
 90 cl = 3
100 ls = &H5555
110 /*
120 repeat
130   line(sx,sy,ex,ey,cl,ls)
140   sx=sx + 4
150   ex=ex - 4
160 until sx > 767
170 /*
180 sx = 767
190 sy = 0
200 ex = 0
210 ey =511
220 /*
230 repeat
240   line(sx,sy,ex,ey,cl,ls)
250   sy=sy + 4
260   ey=ey - 4
270 until sy > 511
280 end
```

見ていると，TVCTRLという命令を見つけることができる。そう，テレビのコントロールだ。これを使うと，ディスプレイテレビを自由にいじくり回すことができてしまうのだ。

リスト9が，そのTVCTRLを使った最も簡単なサンプルである。実行すると画面がいきなりTVに切り換わる。

しかし，ただテレビ画面に切り換えるだけなら，リストを入力するよりキーボードから“SHIFT+.”を入力するほうが早い。これでは実用的ではないので，リスト9にひとひねり加えて，リスト10のようにしてしまおう。

ひとひねり加えただけなのに，リストがいきなり3倍以上の長さになってしまうのはアセンブラの宿命である。アセンブラは計算は得意だが，難しいプログラムを書くのは大の苦手だからだ。

リスト10を実行すると，画面がテレビに切り換わるだけでなく，マウスボタンを押すたびにチャンネルが切り換わるというすぐれものだ。左ボタンを押すと，チャンネル番号はひとつ小さくなり，右ボタンを押すとチャンネル番号がひとつ大きくなる。さらに両方のボタンをいっしょに押すと，プログラムを終了するようになっている。早い話がマウスをテレビのリモコンにしてしまうプログラムなのだ。

このプログラムではTVCTRLのほかに，MS_GETDTというIOCSコールを使っている。マウスの移動量とボタンのON/OFFを調べる命令だ。移動量は無視して，ボタンが押されたかどうかの判定だけを行っている。プログラムの詳しい内容はプログラマーズマニュアルなどの解説書に譲るので，内容を知りたい人は自分でがんばって調べるように。

実はこのプログラムを組んだのはかなり前なので，自分でも解析するのがおっくうなのだ（プログラマの間では「半月前の自分は他人」という格言がある）。幸い68000の解説書は，どれを見ても丁寧にアセンブラの文法を説明してくれているので，やる気になればいくらでもできるだろう。プログラムを本気で作りたい人なら，このくらいのプログラムは，寝ていても読み書きできなくてはいけないのだ。

## 終わりなどない

さて，ここらへんで私の話は終わりであるが，アセンブラの勉強に終わりはないのである。まず，X-BASICの関数のかわりにIOCSコールとDOSコールの使い方について覚え，やがてそれらの速度に不満を持つようになると，次にそれらの高速ルーチンを自前で作るようになる。そして気がつけばバリバリのアセンブラ人間に……，というように深みにハマルいっぽうになりがちなのだ。

私はそんな生き方は避けようと，極力IOCSコールとDOSコールを使ってプログラムを書くようにしている。人間，ときには妥協が必要だ。バリバリのマシン語人間よりも，X-BASICでのほほーんとプログラムを書き，へらへらーと他人の書いたプログラムで遊ぶほうが性に合っている。よく考えたら，私はアセンブラやマシン語という類の言葉が嫌いなのかもしれない。みんなはどうなのかな？　とにかく，気楽にやろうよ。

**リスト8**

```
 1: include          iocscall.mac
 2: .include         doscall.mac
 3:
 4: *         こっちはアセンブラ          * BASICで書くとこうなる
 5:
 6:           move.w   #16,d1            *
 7:           IOCS     _CRTMOD           *screen 2,0,1,1
 8:           IOCS     _G_CLR_ON         *
 9:                                      *
10:           lea.l    linebf,a1         *←ここがBASICには無いところ
11:           move.w   #0,sx             *sx = 0
12:           move.w   #0,sy             *sy = 0
13:           move.w   #767,ex           *ex = 767
14:           move.w   #511,ey           *ey = 511
15:           move.w   #3,cl             *cl = 3
16:           move.w   #$5555,ls         *ls = &H5555
17:                                      *
18: xloop:                               *repeat
19:           IOCS     _LINE             *  line(sx,sy,ex,ey,cl,ls)
20:           add.w    #4,sx             *  sx = sx + 4
21:           sub.w    #4,ex             *  ex = ex - 4
22:           cmp.w    #767,sx           *until sx>767
23:           bmi      xloop             *
24:                                      *
25:           move.w   #767,sx           *sx = 767
26:           move.w   #0,sy             *sy = 0
27:           move.w   #0,ex             *ex = 0
28:           move.w   #511,ey           *ey = 511
29:                                      *
30: yloop:                               *repeat
31:           IOCS     _LINE             *  line(sx,sy,ex,ey,cl,ls)
32:           add.w    #4,sy             *  sy = sy + 4
33:           sub.w    #4,ey             *  ey = ey - 4
34:           cmp.w    #511,sy           *until sy>511
35:           bmi      yloop             *
36:           DOS      _EXIT             *end
37:
38: linebf:
39: sx:       .dc.w    0                 *int sx = 0
40: sy:       .dc.w    0                 *int sy = 0
41: ex:       .dc.w    0                 *int ex = 0
42: ey:       .dc.w    0                 *int ey = 0
43: cl:       .dc.w    0                 *int cl = 0
44: ls:       .dc.w    0                 *int ls = 0
```

**リスト9**

```
1: .include          doscall.mac
2: .include       iocscall.mac
3:      move.l  #5,d1
4:      IOCS    _TVCTRL
5:      DOS     _EXIT
```

**リスト10**

```
 1: .include          iocscall.mac
 2: .include          doscall.mac
 3:
 4:          move.l    #5,d1
 5:          IOCS      _TVCTRL
 6: *
 7: loop:
 8:          IOCS      _MS_GETDT
 9:          and.l     #$0000ffff,d0
10:          move.l    d0,d1
11:          lsr.l     #8,d1
12:          and.l     #$ff,d0
13: *
14:          move.l    d0,d2
15:          add.l     d1,d2
16: *
17:          cmp.l     #$1FE,d2
18:          beq       exit
19: *
20:          tst.l     d2
21:          beq       loop
22: *
23:          tst.b     d0
24:          bne       cup
25:          move.l    #$0c,d1
26:          bra       chset
27: *
28: cup:     move.l    #$0b,d1
29: chset:   IOCS      _TVCTRL
30:          bra       loop
31:
32: exit:    move.l    #$1d,d1
33:          IOCS      _TVCTRL
34:          DOS       _EXIT
```

吾輩はX68000である　番外編

# デバッガにて理解されたし

Izumi Daisuke 泉　大介

アセンブラを修得しようとしたが，どこから手をつけていいかがわからない。そんな諸兄には吾輩は，まずデバッガでプログラムを覗きみて，アセンブラの動きを追ってみるところから始めるという学習法もいいのではないかと提案する。

吾輩がもって生まれた多彩なハードウェアの能力を，いかんなく発揮したアクションゲームの数々が市場を賑わせている。65536色を同時に発色するグラフィックとスムーズなキャラクタ移動を簡単に実現するスプライトで構成される異次元の世界。そして，諸兄を否応なくこの異次元の世界へとトリップさせる８重和音のFM音源，効果音を再生するPCM。さらには，半透明機能からラスタスクロールまで，最近のアクションゲームは吾輩の能力を極限まで引き出して，諸兄を夢の世界へといざなっている。

これらのゲームが，「かくあれかし」と願うことによってではなく，地道に命令を１つひとつ組み立てて制作されていることを諸兄はご存知であろう。スプライトをAからBへ動かせ。No.4のスプライトとNo.5のスプライトが重なれば，両者を画面から消去し爆発パターンを表示しろ。こういった命令が調和し，ゲームというひとつの世界を作り上げているのだ。人間がこれらの命令を吾輩に伝えるための手段として作り出したのがBASICであり，C言語であり，FORTRANであり，……そう，様々なコンピュータ言語なのである。

吾輩はコンピュータである。蛋白質皮膜を使った有機分子ではなく，半導体を使った無機分子で構成される古典的なコンピュータである。DNAやRNAではなく，酵素でもなく，ましてや細胞の興奮レベルも関係ない単純な矩型をした電気パルスによって駆動されている。これが吾輩の中を駆け巡る命令の唯一とりうる形態であり，吾輩の理解できる唯一の言語なのである。電気信号のあるなし，ON/OFFを数字の１と０に対応させ，人は吾輩に命令を与える。この延々と続く１と０の羅列が，俗に「マシン語」と呼ばれるものである。

吾輩が理解できる言語が電気信号，ひいてはマシン語ただひとつであるという事実は諸兄を少々混乱させるかもしれない。いまやすっかりメジャーな言語に成り上がったC言語や，電気仕掛けのコンピュータの歴史の曙より脈々と受け継がれてきたFORTRANをコンピュータは理解できないのだろうか，という疑問もお持ちのことと思う。

結論からいえば，吾輩はこれらの言語を理解することはできない。どんなに簡単なC言語の命令も，吾輩に直接命令を与えることはできないのである。これらの言語でプログラムを作成できるのは，C語やFORTRAN語をマシン語に変換するためのプログラムであるコンパイラ（当然これはマシン語で書かれている）があるが故である。吾輩は，コンパイラがせっせと変換してくれたマシン語を見て初めて，プログラムに書かれていることを理解し実行することができる。

BASICのほうは若干事情が異なっており，BASIC語がマシン語に変換されるわけではない。BASIC語を解釈実行するX-BASICというマシン語のプログラムが動いているだけである。吾輩はX-BASICを動かしており，そのX-BASICが諸兄のプログラムを動かしているのだ。だからといって，吾輩がBASIC語を理解していると考えるのは間違いである。吾輩がマシン語で書かれたCコンパイラを動かしてC語をマシン語へ変換するからといって，C語を理解していると考えるのも間違いである。

ちょうどこれは，諸兄の精神活動が脳のニューロンの興奮レベルとは別の次元で機能しているのと同じことである。幾多の階層をなすニューロンのネットワークはそれぞれに興奮し，電解質を分泌して様々な情報を処理しているが，だからといって個々のニューロンが「今日は暑いなぁ」などと考えているわけではない。ニューロンのネットワークを流れる情報の総体が「暑い」と感じるわけでもない。それは別の次元に属するのである。

マシン語を操ることは，諸兄が特定の細胞を意識的に興奮させることに相当するといえるだろう。眼球は上にも下にも移動させることができるが，バカボンのパパのように左目を上に，右目を下に同時に移動させることはできない。ハードウェアとしてその能力を持ってはいても，高次の意識はそれを命令することができないのである。同様に吾輩の持つハードウェアの能力を極限まで引き出すことができるのは，マシン語ただひとつであるといえる。

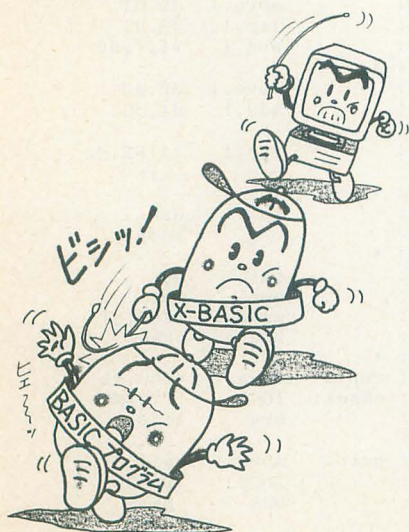
## マシン語，その傾向

数字の１と０の羅列で表現される電気信号のON/OFFが，吾輩が唯一理解することのできるマシン語だと説明したが，

```
01110000 00000001
01001110 01110101
```

のように，２進数を羅列するのは人間にとって非常に読みづらいものである。そこで，２進数４桁で0〜15までの数を表現できることに注目し，上の２進数を，

```
70 01
4E 75
```

と16進数で表現する方法が考案され，一般に採用されている。

諸兄は上の数字を見て，これが何を指示しているのかおわかりだろうか。吾輩にとっては一目瞭然の数字ではあるが，この意味するところを覚えておくのは人間にとって非常に面倒なものであるらしい。そこで，数字をその機能を表す英字を使って表現する方法が考案された。この方法によると上の数字は，

```
moveq #1,d0
rts
```

と表すことができる。moveqはデータを移動するmove命令のバリエーションのひとつである。ここではD0に1をセットすることを指示している。続くrtsはサブルーチンから帰ること(return from subroutine)を指示している。どうせなら完全な英単語にしたほうがわかりやすいのではなかろうかと思うのだが，吾輩のあずかり知らぬ理由によって英単語のハナモゲラが採用されている。

ここまでくると，ようやく人間にも意味の通る表現と見なすことができるらしく，吾輩の母国語たるマシン語と1対1に対応するこのハナモゲラ語は，「アセンブリ言語」というたいそうな名前をもらうに至った。当然吾輩はアセンブリ言語を直接理解することはできないので，アセンブラなるプログラムを利用してアセンブリ言語をマシン語に変換するという作業が必要である。とはいうものの,「マシン語と1対1に対応している」という事実を重視して，アセンブリ言語をマシン語と呼ぶ場合も少なくない。電気信号のON/OFFを数字の1と0に置き換えた時点で，すでに純然たるマシン語とはいえなくなっているので，それを16進数に直すのも，さらにもうちょっと手を加えてハナモゲラにするのも大差ないということなのかもしれない。実際，現在では16進表記のマシン語を読めるということは，ソンケーの眼差しで見られるという以外に大して意味はないといってもよい。マシン語でプログラムするということは，アセンブリ言語でプログラムするということを意味するようになっている。

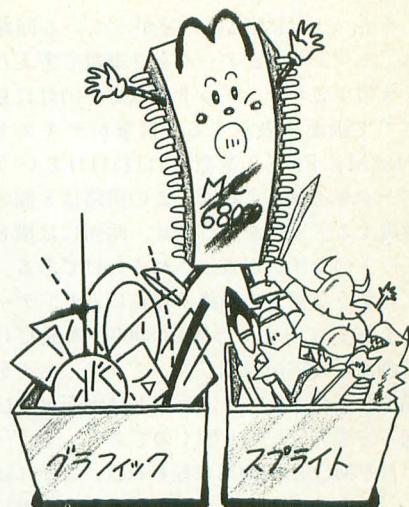
マシン語は，体の中を直接触られるという面白くない事態を吾輩にもたらすという点を除いても，次のような問題を抱えている。ひとつは，その命令1つひとつがあまりにも微細な命令であるということだ。これを人に例えれば，細胞のひとつを興奮させることができるというレベルのものである。このような命令を組み合わせてまとまったプログラムを作り上げるということは，あるエネルギーをもったフォトンに興奮する視神経細胞を組織して幻覚を見せるに等しい業である。マシン語のプログラマが尊敬されるわけもここにある。マシン語プログラム師を目指している奇特な諸兄に忠告しておく。掛け算の演算子をひとつ作るのに，何百もの命令を組織しなければならないからといってメゲてはいけない。それがミニマムなレベルに侵入して吾輩を自由に操ろうという，その志の代償なのだ。

マシン語はまた，その動作を途中で簡単に中断させることができないという問題点をも持っている。通常，X-BASICではプログラムの動作が変な場合は，すぐさまBREAKキーで動作を中断させることができるが，マシン語では吾輩の頭にまで腕を伸ばし(ツインタワーの場合)，ヤワであまり耐久性が高くはなさそうなINTERRUPTボタンを押さなければならない。これはプログラムを超高速に実行できることの代償と受け止められたい。

これでプログラムの実行を中断することはできるが，X-BASICのようにすぐさま変数の内容を調べ，プログラムに検討を加えることはできない。対話性はよくないのである。とはいえこれでも，プログラムを少し間違えていただけでシステムをぶっ壊し，あさっての世界に行ってしまっていたZ80時代に比較すると，問題のプログラムだけを中断して元の状態に復帰できるだけありがたいと思っていただきたいものだ。少なからぬ進歩を遂げているのである。

そこで

デバッガの登場である。これまでに何度か，吾輩の機能を紹介するのに使用しているのでご存知の諸兄も少なくあるまい。デバッガは，かように面倒なマシン語のプログラムのデバッグを，少しでも容易にしようと用意されたプログラムである。その最も大きな目的は，任意の場所でマシン語プログラムの実行を中断できるようにすることと，そのときの変数の様子を眺めることができるようにする点である。

マシン語プログラム師を目指している諸兄ならば，当然福袋ver.2.0あるいはCコンパイラをお持ちのことと思う。デバッガはこれらの中に収められている。また，まだそこまで本気になっていない諸兄も，Oh!Xの愛読者ならば1991年1月号の付録ディスクをお持ちのことと思う。この中にも同等のものが収められている。いずれを使用されても構わないが，デバッガのコマンドがバージョンの古いものと新しいものでは若干異なっているので注意されたい。ここでは，付録ディスクのデバッガを例に取り上げることにする。

## MC68000の世界

デバッガにいく前に，まずマシン語が扱う世界について触れておこう。吾輩の頭脳はMC68000と呼ばれるマイクロプロセッサである。この頭脳は，データをあちらからこちらへ移動したり，簡単な加減乗除の演算を行ったり，データを2進数で考えたときに特定の桁が1になっているかどうかを調べたりする機能を持っている。さらに，メモリにデータを書き込んだり，メモリからデータを読み出す機能も備えている。スプライトをAからBへ動かしたり，グラフィックを表示する機能……，これは持っていない。FM音源を鳴らす機能……，これも持っていない。諸兄が吾輩の欠くべからざる機能だと，ゲームには不可欠の要素だと考えている諸々の機能を実現する能力は，MC68000にはないのである。

だからといって，どうかがっかりしないでいただきたい。吾輩が扱うことのできる言語がマシン語ただひとつであり，その吾輩が立派にゲームをこなしているからには，なんらかの秘訣があるはずなのだ。

MC68000はメモリを操作することができる。このメモリというやつがくせものなのである。メモリというと，電卓のメモリ機能のようにデータを一時的に保存しておき，然るべきときに再び取り出して利用するための貯蔵庫のようなものを連想されるだろう。吾輩の持っているメモリも，基本的にはこれと同じように動作する。ところがなかには，吾輩がセットしたデータをこっそり盗み見し，そのデータを別の目的に使用するための回路がつながっているメモリが存在する。テキストVRAMと呼ばれる

メモリもこの一派である。

テキストVRAMにつながっている回路は，セットされたデータを2進数で考えたときの1と0を，ドットの点灯と消灯に見立てて画面に表示する。吾輩がテキストVRAMに$FF_H$(2進数で11111111)というデータをセットすれば，この回路は8個の連続するドットを点灯させ，画面には横8ドットのラインが表示されるわけである。このような裏方の回路の存在により，データを移動するというMC68000の機能だけで画面に文字が表示でき，グラフィックを描くことができ，スプライトは画面を飛び回って爆発音が鳴り響くのである。

目をMC68000内部に転じれば，ここにはレジスタと呼ばれるメモリがある。一般に使用するのはデータレジスタD0～D7，アドレスレジスタA0～A7の16個で，数は決して多くない。しかしながらその転送速度は一般のメモリを遙かにしのぐ。32ビットのデータをメモリからメモリに移動するより9倍も速く，同じデータをレジスタからレジスタに移動することができるのである。

そして，もうひとつ，忘れてはならないレジスタが存在する。フラグレジスタである。このレジスタはMC68000が現在どのような状態にあるかを示すためにある。MC68000は，計算の結果がゼロになれば，「ハイ，ゼロになりました」と旗を立てる。計算結果が指定された桁に収まらず，繰り上がりが発生した場合は，「ハイ，繰り上がりました」と別の旗を立てる。上の桁から借りてこなければ引き算できない場合には，「ハイ，借りてきました」と旗を立てる。このゼロになったときに立てられる旗や，繰り上がりがあったときに立てられる旗などの集合体がフラグレジスタなのである。

MC68000は「D0がD1より大きければ」というような条件を判定することはできない。「旗が立っているかどうか」を判定することしかできないのである。「D0がD1より大きければ」という条件を判定したければ，

1) D0－D1を計算
2) フラグを調べる

という2段構えで臨む必要がある（実際には，引き算を行わずにD0－D1を調べる命令を使用する)。フラグレジスタの重要性がおわかりいただけただろうか。同時にこれは，ミニマムなレベルのプログラミングというものが，どのようなものなのかを示す例でもある。

MC68000は，メモリからデータを取り出し，レジスタを使いながら様々な計算を行う。そして，その結果を再びメモリに書き込む。あるメモリはメモとして機能し，あとで再びその値が必要になったときのためにデータを保存する役目を負う。またあるメモリは，外部回路によってデータを盗み出され，メッセージや感動のグラフィックとなって諸兄の注意を喚起する。かくして，レジスタとメモリしか扱えないMC68000の閉じた世界は，諸兄のアイデアによって様々な開かれた世界と交歓するのである。

**図1　テキストVRAMをダンプ**

```
-d e00000
00E00000  0000 0000 0000 0000 0000 0000 0000 0000    ................
00E00010  0000 0000 0000 0000 0000 0000 0000 0000    ................
00E00020  0000 0000 0000 0000 0000 0000 0000 0000    ................
00E00030  0000 0000 0000 0000 0000 0000 0000 0000    ................
00E00040  0000 0000 0000 0000 0000 0000 0000 0000    ................
00E00050  0000 0000 0000 0000 0000 0000 0000 0000    ................
00E00060  0000 0000 0000 0000 0000 0000 0000 0000    ................
00E00070  0000 0000 0000 0000 0000 0000 0000 0000    ................
```

## デバッガ事始め

デバッガは，メモリやレジスタの内容を見るために，あるいはメモリやレジスタに特定のデータをセットするために，様々なコマンドを持っている。これらのコマンドは，

```
X68k Debugger v ……
-■
```

のように，デバッガのプロンプトである「－」が表示されカーソルが点滅している状態で，hとタイプしリターンキーを押せば見ることができる。

メモリに入っているデータを見るには，この中の「D」コマンドを使用する。逆にメモリにデータをセットするには「ME」コマンドを使用する。大文字を使用しても小文字を使用しても構わない。X68000は標準で1あるいは2Mバイトのメインメモリを持っており，諸兄の心掛け次第で最大12Mバイトまで実装可能である。したがってメモリを操作するときには，メモリのどこを対象とするのか指示してもらわねばならない。メモリには場所を示すためのアドレスと呼ばれる番号が振ってあるので，これを利用することになる。たとえばテキストVRAMはアドレス$E00000_H$から始まっているので，この内容を見るには，

```
-d e00000
```

とすればOKである。これで図1のように$E00000～E0007F_H$までの128バイトが画面に表示（メモリダンプ）される。表示されるデータが0ばかりなのは，これが画面の一番上のラインに相当しているからである。上下の文字がくっついてしまわないように，半角文字の一番上のラインは空白になっているので，たとえ最初の行に文字が表示されていても0になるのである。

```
-d e00400
```

くらいで，文字の片鱗が眺められるであろう。1ラインは$80_H$バイトなのでこれは9ライン目に相当する。試してみていただきたい。デバッガではこのようにいきなり数値を並べると16進数を意味するようになっている。10進数で指示したければ，数字の前に「¥」をつければいい。2進数なら「_」である。データをセットする場合も同様にアドレスを指定して行う。

レジスタへのデータセットには「X」コマンドを使用する。単に

```
-x
```

とすれば，図2のように現在のすべてのレ

### 付録ディスクのデバッガの使い方

1991年1月号付録の「謹賀新年PRO-68K」は4枚のディスクを生成するようになっている。このうちのDisk4に目的のデバッガは入っている。Disk4をセットして電源を入れるとビジュアルシェルが起動するが，マウスの右ボタンを押すと現れるポップアップメニューから「Exit」を選択して，まずはビジュアルシェルを終了されたい。画面表示は，

A>

に変わったはずである。ここで，

A>a:¥sx¥tool¥db

とキーをタイプしてリターンキーを押せば，デバッガが起動する。

デバッガを起動するのに，「a:¥sx¥～」とタイプするのが面倒な諸兄は，

A>path a:¥sx¥tool

とタイプしてリターンキーを押されたい。これで，

A>db

とするだけでデバッガが起動するようになる。デバッガはAドライブのSXディレクトリのTOOLディレクトリの中に入っており，本来は最初の例のようにデバッガのある場所を指定して起動しなければならないのだが，pathを使用すると指示したディレクトリから自動的に起動するようになるのである。

ジスタの状態が表示される。行頭にDと表示された行がD0～D7, 行頭にAと表示された行がA0～A7にセットされているデータである。フラグレジスタはXフラグ，Zフラグのように分けて列挙されている。レジスタにデータをセットするには，

```
-x d0 30
```

というふうに行う。これでD0レジスタに$30_H$をセットすることができる。もっとも，諸兄が直接レジスタを触ることはまずないだろうが。

さて吾輩を駆動している問題のマシン語プログラムであるが，これはメモリにデータとして収められる。MC68000の扱う対象がレジスタとメモリなのだから当然だ。

たとえば，$100000_H$から「me」コマンドを使って，

```
-mes 100000
00100000 00 ：70
00100001 00 ：01
00100002 00 ：4e
00100003 00 ：75
00100004 00 ：.
```

とデータをセットしていけば，それがプログラムとなるのである。「mes」は「me」コマンドに「1バイトずつデータを書き込む」ことを指示する「s」を付加したものである。コマンドの中で［<size>］のオプションのあるコマンドは，このようにsを付加することでバイト単位，wを付加することでワード単位，lを付加することでロングワード単位と指示できる。最後の行にピリオドを入れているが，これはデータセットの終了を意味する。

少々驚かれた諸兄が在るかもしれない。「これではメモリに入っているデータが，純然たるデータなのか，プログラムなのかわからないではないか」と思われることだろう。まったくそのとおりである。吾輩X68000にとって両者はまったく区別できないのだ。プログラムとして実行しろといわれれば，吾輩は「7001 4E75」をプログラムとして実行する。アドレス$100000_H$から1ロングワードのデータを取り出せといわれれば，吾輩は「70014E75」というデータを取り出す。「おや？ 7001 4E75はプログラムだぞ。なにかおかしいのではないのかな」などと考えることは金輪際ない。したがって諸兄がなにかの間違いでデータ列を実行させてしまった場合，吾輩は文句もいわずデータを取り出してはそれをプログラムと見なし，ありもしない命令を実行し続け，挙げ句の果てに奇数アドレスからワードデータを取り出そうとして「アドレスエラー」を起こしたり，実装されていないメモリからデータを取り出そうとして「バスエラー」を起こしたり，いよいよ最悪の場合にはあさっての世界に行ったきり帰ってこなくなったりするのである。

話が横道に逸れてしまった……。先の「mes」コマンドの例を見て，ちょっとマシン語プログラミングの世界を覗いてみてやろうと軽く考えている諸兄の中には，重苦しい気持ちを抱いている方がいらっしゃるかもしれない。メモリに直接16進データをセットしていかなければならないというのは，あまりに面倒な作業といえる。だが，ご安心いただきたい。この例は，マシン語の一般教養として紹介しただけである。手作りマイコンの時代ならいざ知らず，この軽薄短小の現在にあってこんなことをやろうという奇特な御仁を探すのは難しい。

### ●A,ANコマンド

デバッガに備わっているA，ANの2つのコマンドは，アセンブリ言語をマシン語に変換するコマンドである。

```
-an 100000
```

のようにアドレスを指定してANコマンドをタイプし，リターンキーを押すと，

```
00100000 ■
```

と画面に表示される。ここで，

```
00100000 moveq #$01,d0
00100002 rts
00100004
```

と命令を打ち込んでいけば，あら不思議。ちゃんとマシン語に変換されて格納されるのである。疑い深い諸兄は，先の「D」コマンドでメモリをダンプされるとよかろう。$100000_H$から「70 01 4E 75」とデータが並んでいるのを確認できる。そう，DB.Xは簡単なアセンブラを内蔵しているのである。ここでも終了にはピリオドを使用する。

「A」コマンドのほうは，これから命令を書き込もうとしているアドレスだけでなく，そこに入っているデータを命令だと見なして表示する点が異なっているだけである。いずれを使用しても結果は変わらない。

### ●Lコマンド

メモリに入っているデータを命令だと見なして，アセンブリ言語で表示するのがこのLコマンドである。もちろんAコマンドやANコマンドでセットした命令を眺めることも可能である。試しに先にセットした命令を見てみると，

```
-l 100000 100002
00100000 moveq #$01,d0
00100002 rts
```

と表示される。

### ●Pコマンド

A，AN，Lコマンドがあれば，アセンブラがなくともマシン語を試してみることはできるわけだが，メモリを諸兄が好き勝手に使うことはできない。諸兄が命令を書き込もうとしているアドレスにはデバッガがあるかもしれないし，あるいは吾輩にとって重要な情報が書き込んであるかもしれない。

そこでこのPコマンドは，

```
-p
debug program from $0007C0B0
user  program from $000A9A90
```

のように，どのアドレスから諸兄がプログラムを作っていいかを表示するコマンドである。ここではデバッガが$7C0B0_H$から始まっており，$A9A90_H$からプログラムを作成してもいいことが示されている。もちろんこのとき表示されるアドレスは，諸兄のマシンの使用状況によって異なってくるので注意されたい。ここまで便宜上$100000_H$に命令を書き込んできたが，メインメモリを1Mバイトしか搭載していないX68000ではここにはメモリはない。Pコマンドで表示されるアドレスに命令を書き込まれたい。

## DB.Xはマシン語インタプリタ

はじめのほうでX-BASICは，BASIC.Xというマシン語プログラムが，BASIC語で書かれたプログラムを解釈実行しているのだと説明した。このようなプログラム実行機構はインタプリタと呼ばれている。

向学心旺盛な諸兄の中には，interpreterという単語を辞書で調べた方もいらっしゃろうかと思う。訳語は「通訳」である。BASIC語のひとかたまり（式ひとつ，コマンドひとつ）ごとに解釈実行し，再びBASIC語のプログラムを眺めて次を解釈実行するという作業形態が，通訳を介した会話の様子に似ているために名づけられたのであろう。

図2　Xコマンドでレジスタを眺める

```
-x
PC=000A9A90 USP=000881D4 SSP=000067F2 SR=0000 X:0  N:0  Z:0  V:0  C:0
D  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 0000000 0000000 000881D4
ori.b           #$00,D0
```

さらに知識欲旺盛な諸兄はcompilerを引いてみて，どうもぴったりこないと感じられたことだろう。訳語は「編集者」である。元の英語が意味するものは種々のpile(集積物)をcom(一緒に)して処理する人である。C語のプログラムを一括処理するため，こう名づけられたのであろう。アメリカ人はかようなイメージを「編集者」に抱いているものらしい。

デバッガDB.Xは，マシン語のプログラムの命令を1つひとつ実行できるという，まるでインタプリタのような能力を持っている。また，これがため，マシン語を始めようという諸兄にはまたとない入門の道具となりうる。通常ならあっという間に実行されてしまう命令を，1つひとつ吟味しながら実行し，そのときのレジスタ変化やフラグ変化を追いかけていくことが可能なのである。その便利なコマンドはTコマンドという。

**●Tコマンド**

```
00100000    move.w   #'@',d0
00100004    move.w   d0,-(sp)
00100006    _putchar
00100008    addq.l   #2,sp
```

というプログラムを例にその様子をご覧に入れよう。

_putcharというのは，画面に文字を表示するDOSコールである。ミニマムレベルのプログラミングということでさんざん脅かしたが，実際にはプログラムを1から10までミニマムな命令の組み合わせで作る必要はない。ある程度まとまった機能を持つ(とはいってもやはり小さな部品程度の機能なのだが)マシン語プログラムが，IOCSコール，DOSコールとして用意されているのである。ここではそれを利用している。

DOSコールを使用するときに必ずついて回るのが，スタックという概念である。スタックは図3のようにデータを積み重ねて保存しておくものであり，最初に積んだデータは最後に取り出されることになる。MC68000がレジスタとメモリしか扱えない以上，このスタックも当然メモリ上に作成される。現在スタックの一番上がどこにあるのか，すなわち，スタックトップのアドレスはどこなのかを保持しているのがsp(スタックポインタ)と呼ばれるレジスタである。MC68000では，A7レジスタがこのスタックポインタの役割を果たしている。

図3をもう少し詳しく説明しておこう。データは，まずスタックポインタを必要なだけ（ワードデータなら2バイト，ロングワードデータなら4バイト）小さくしてから書き込まれる。逆に取り出すときは，データを取り出してからスタックポインタが大きくされる。かくして，先入れ後出しのデータ貯蔵庫が完成するわけである。

DOSコールの前にある「move.w d0,-(sp)」について触れておこう。スタックポインタにセットされているアドレスにD0レジスタのデータを書き込むには「move.w d0,(sp)」とすればいい。データを書き込む前にスタックポインタを小さくするには，「(sp)」の代わりに「-(sp)」を使う。逆に，データを取り出してからスタックポインタを大きくするには「(sp)+」とする。つまりスタックからD0レジスタにデータを取り出すには，「move.w (sp)+, d0」とすればいいことになる。DOSコールは，このように必要なデータをスタックに積んでから使用するようになっている。

勘のいい諸兄は，「spとはA7レジスタのことだった。-(a7)や(a7)+ができるなら，同じアドレスレジスタであるA0やA6でもスタックを作ることができるのではないだろうか」と思われるかもしれない。そのとおりである。MC68000はA7レジスタをスタックポインタとして使用するが，これとは別に諸兄がA6レジスタをスタックポインタとする自分のプログラム用のスタックを作成しても構わない。いうまでもないことだが，スタック用のメモリは諸兄が自分で用意する必要がある。本当はA7レジスタで示されるスタックも用意しなければならないのだが，ここではデバッガの持っているスタックを利用することで省略している。

このプログラムは，AやANコマンドで作成可能である。Pコマンドで表示されたアドレスに作成してみていただきたい。終わったらLコマンドで確認してみるのをお忘れなく。spと入力したはずがA7と表示されるので戸惑われるかもしれないが，それはデバッガの都合ということでご容赦願いたい。ではTコマンドにとりかかろう。図4をご覧いただきたい。

Tコマンドは，実行を開始するアドレスを「＝アドレス」の形で与えて使用する。最初の命令，

```
move.w   #'@',d0
```

が実行されるとすぐさま，DB.Xはレジスタの状態を表示して入力待ちになる。「@」のアスキーコードである$40_H$がD0レジスタにセットされているのを確認できるだろう。レジスタ内容の表示の後ろには，次に実行しようとしている命令が表示されている。

2回目以降は，単に「t」とだけ入力していけばいい。

**図3 スタックの性質**

**図4 Tコマンドの実際**

```
-t=100000          ← 100000Hからトレース開始
PC=00100004 USP=000881D4 SSP=000067F2 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  00000040 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D4
move.w  D0,-(A7)                  :000881D2(0000)
  ↑ 次の命令を表示して停止する

-t                 ← 再びトレース
PC=00100006 USP=000881D2 SSP=000067F2 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  00000040 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D2
_PUTCHAR                                                              ↑
                                                        スタックポインタが変化した
-t                 ← トレース。PCの前に@が出現
@PC=00100008 USP=000881D2 SSP=000067F2 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  0001001E 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D2
addq.l  #2,A7

-t
PC=0010000A USP=000881D4 SSP=000067F2 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  0001001E 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D4
ori.b   #$00,D0           ← 意味のない命令                              ↑
                                                        スタックポインタが元に戻った
```

move.w　d0,-(sp)

でスタックに$40_H$が積まれる。実際にスタックに積んでいるところは表示されないが，スタックポインタであるA7レジスタの値が，２つ小さくなっているのがおわかりだろうか。ワードデータを積んだので２つ小さくなったのである。もしなんなら，

-d .a7

としてスタックを表示してみられるとよい。シンボル（ここではレジスタ名）の前にピリオドをつけると，その内容を入力したのと同じことになるので，これは，

-d 881d2

という意味になる。メモリダンプの先頭には，$0040_H$というデータが表示されているはずである。

続く「t」コマンドで，

_putchar

が実行される。このDOSコールは，スタックに積まれたデータをASCIIコードだと見なして文字表示を行うので，

@PC＝00100008 USP＝……

のように「@」が表示されることになる。これでDOSコールは終了したが，スタックポインタは$881D2_H$のままである。自動的にスタックポインタが戻されるということはないのだ。そこで最後に，

addq.l　#2,sp

でスタックポインタを元に戻す必要がある。ここでは，スタックポインタに2を加えて元に戻している。DOSコールを使用する場合には，スタックポインタの戻し忘れに注意されたい。こういった細かなところまで神経を配る必要があるのが，マシン語プログラミングなのである。

**●フラグ，その変化**

MC68000は通常アドレスの小さいほうから大きいほうへ順序よく命令を実行していくが，条件によってこの実行順序を変化させることができるようになっている。条件というのは，前述したように「フラグが立っているかどうか」の判定である。

```
00100000　move.l　#1,d0
00100006　subq.l　#1,d0
```

というプログラムをＴコマンドで追いかけてみる。図５である。最初の，

move.l　#1,d0

によってD0に１がセットされるが，続く，

subq.b　#1,d0

で０になっている。このとき，「Z:0」と表示されていた場所が，「Z:1」と表示し直されている点に注意されたい。計算の結果が０になったので，ゼロフラグが立ったのである。図５ではさらにここから１を引いている。

**図5　フラグセットの様子**

```
-t=100000
PC=00100006 USP=000881D4 SSP=000067F2 SR=8010 X:1  N:0  Z:0  V:0  C:0
D  00000001 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D4
subq.l  #1,D0             ← D0(1がセットされている)から1を引くと,

-t                                                    ↓ ゼロフラグがセットされる
PC=00100008 USP=000881D4 SSP=000067F2 SR=8004 X:0  N:0  Z:1  V:0  C:0
D  00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D4
subq.l  #1,D0             ← さらに1を引くと,
                                         下の3つのフラグがセットされる
-t                                          ↓    ↓              ↓
PC=0010000A USP=000881D4 SSP=000067F2 SR=8019 X:1  N:1  Z:0  V:0  C:1
D  FFFFFFFF 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D4
ori.b   #$00,D0
```

その結果，Ｘ，Ｎ，Ｃの３つのフラグが立っていることが確認できよう。

Ｃフラグはキャリフラグと呼ばれていて，計算の結果，繰り上がりや借り入れが発生したことを示している。０から１を引くことはできない。そこで上の桁から１を借りてきて，$100000000_H$から１を引いているのである。結果は$FFFFFFFF_H$となり，借りてきた印としてキャリフラグが立っている。そしてＸフラグにもキャリフラグが反映され，「X:1」となるのである（ただし，命令の中にはＸフラグにＣフラグが反映されないものも存在する）。

この計算結果には面白い側面がある。吾輩たちコンピュータが負の数を扱えないというのはまことにもって不便だと人は考えたに違いない。$FFFFFFFF_H$に１を加えると（キャリが発生して）０になることに注目し，疑似的にこの$FFFFFFFF_H$を－１だと見なす，「２の補数表現」と呼ばれる手法が考え出されたのである。そして，$00000000_H$～$7FFFFFFF_H$を正の数，$FFFFFFFF_H$～$80000000_H$を負の数と見なすことになった。最上位桁に注目すると，0～$7_H$は２進数で$0000_B$～$0111_B$，8～$F_H$は２進数で$1000_B$～$1111_B$となるので，数値を２進数で考えたときの最上位桁が０なら正の数，１なら負の数ということもできる。

そしてMC68000は，「データが仮に符号つきで表現されていると考えたときにどうなるか」を反映するフラグを持っている。Ｎ（Negative：負の数）フラグは，計算結果が負の数と見なされる場合にセットされる。図５でも，D0から１を引いて$FFFFFFFF_H$になったときにＮフラグがセットされているのが確認できよう。

残るフラグはＶフラグだが，これはデータの計算結果が，符号つきの表現では表せない値となってしまった場合に立てられる。つまり，$7FFFFFFF_H$に１を加えたり，$80000000_H$から１を引いた場合がそれである。各自で実験されたい。

**図6　数値の比較に関する条件**

| 符号無 | 符号付 | 意味 | 条件 |
|---|---|---|---|
| hi | gt | Greater Than | A＞B |
| cc | ge | Greater Equal | A≧B |
| eq | eq | EQual | A＝B |
| ne | ne | Not Equal | A≠B |
| ls | le | Less Equal | A≦B |
| cs | lt | Less Than | A＜B |

**●フラグの判定方法**

このフラグは，分岐命令(branch)と一緒に利用される。分岐命令は「Bcc」という形をしており，「cc」で条件を指定するのである。たとえば「キャリフラグが立っていたら分岐する」という命令は，「bcs(Branch if Carry Set)」となる。「キャリフラグが立っていなければ分岐する」という命令は，「bcc(Branch if Carry Clear」とする。この他にも，Ｎフラグを判定するpl(PLus)とmi(MInus)，Ｖフラグを判定するvs(oVerflow Set)とvc(oVerflow Clear)といった条件が用意されている。

ゼロフラグの判定は少し変わっていて，ゼロフラグが立っているなら条件としてeq(EQual)を，立っていないならne(Not Equal)を使用するようになっている。これは，A－Bを計算したときに，A＝Bならゼロフラグが立つことに起因している。同様の命名方法で名づけられた条件として，A＞B(キャリクリアかつゼロクリア)を判定するhi(HIgh)，A≦B(キャリセットあるいはゼロセット)を判定するls(Lower or Same)がある。これらは２つのフラグを同時に判定する方法としても利用されたい。

大小判定に起因して命名された「条件」は，符号つきの数値比較ではさらに徹底した様相を呈する。図６を参照されたい。こちらはフラグが立っているかどうかなど一顧だにせず，素直に条件を利用したほうが混乱がなくていいだろう。ついでに，上記のhi，lsなどの条件を大小判定に使う場合

の利用法も併せて示しておいた。もちろん，$00000001_H$－$FFFFFFFF_H$を符号つきで計算しているのかどうかは諸兄のみ知るところで，吾輩，ひいてはMC68000はまったく関知していない。それを符号なしの計算として条件判定をしたければ「符号無」のシリーズを，符号つきの計算として判定したければ「符号付」のシリーズを利用すればいいだけである。

**●ループは回る**

Bcc命令を使えば，画面にn個の@を描くなどという処理を簡単に記述することができる。

```
00100000   moveq   #2,d1
00100002   move.w  #'@',-(sp)
00100006   _putchar
00100008   subq.b  #1,d1
0010000A   bne.b   $00100006
0010000C   addq.l  #2,sp
```

というプログラムを追いかけてみよう。このプログラムのエッセンスは$100008_H$と$10000A_H$の2つの命令にある。D1レジスタから1を引き，結果がゼロでなければ$100006_H$に戻って再びプログラムを実行するのである。結果は図7である。フラグ変化と実行される命令を対比させながら眺めていただきたい。

**●ブレイクポイントとGコマンド**

命令を1つひとつ実行して眺めることができるTコマンドの便利さは実感していただけたことと思うが，プログラムがある程度以上大きくなると，この作業は非常に煩わしく感じられるようになってくることであろう。実際，目的の場所までは一気に実行し，そこから命令を1つひとつ確かめていくことができなければ実用にはならないといってもいい。そのための命令がDB.Xに用意されている。

図8-1のようなプログラムを考えてみることにしたい。このプログラムでは，「画面に@を表示する」という機能を持ったサブルーチンを$100010_H$に用意し，$100002_H$でそのサブルーチンを利用している。先程のD1を減じてからBccでループという部分は，「dbra」という命令に置き換えてある。

これは，指定されたレジスタの値を（ワードサイズで）1減じ，負の数にならなければループする。すなわち，

```
subq.w  #1,d1
bpl.w   $100002
```

を1命令にしたものである。また最後の，

```
_exit
```

はプログラムを終了するためのDOSコールである。

Gコマンドは，通常の速度でプログラムを実行するコマンドである。画面にレジスタの状態が表示されることもない。このままだとプログラムを最後まで実行してしまうので，中断したい場所をブレイクポイントとして指定して使用する。図8-2がブレイクポイントを指定しているところで，ここではdbra命令を実行するところで止まるように設定してある。

図8-3をご覧いただきたい。ブレイクポイントまで一気に実行され，レジスタが表示されてプログラムが中断されているのを確認いただけると思う。ここでは再びGコマンドを使用しているが，Tコマンドでプログラムを追いかけ，適当なところで再びGコマンドを使用しても構わない。問題の場所だけを丁寧に吟味し，そのほかの場所はカッ飛ばして実行するのにGコマンドとブレークポイントのペアは最適である。

**●Sコマンドもあるでよ**

DB.Xではなく正当にアセンブラでプロ

**図7 ループを試す**

```
-t=100000                  ← D1にループ回数2をセット
PC=00100002 USP=000EEB64 SSP=000BBCB4 SR=8000 X:0  N:0  Z:0  V:0. C:0
D  00000000 00000002 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000EEB64
move.w  #$0040,-(A7)               :000EEB62(0000)
-t
PC=00100006 USP=000EEB62 SSP=000BBCB4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  00000000 00000002 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000EEB62
_PUTCHAR
-t
@PC=00100008 USP=000EEB62 SSP=000BBCB4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  0001001B 00000002 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000EEB62
subq.b  #1,D1              ← D1から1を引く
-t
PC=0010000A USP=000EEB62 SSP=000BBCB4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  0001001B 00000001 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000EEB62
bne.s   $00100006          ← ゼロフラグが立たなければ100006Hへ分岐
-t
PC=00100006 USP=000EEB62 SSP=000BBCB4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  0001001B 00000001 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000EEB62
_PUTCHAR
-t
@PC=00100008 USP=000EEB62 SSP=000BBCB4 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  0001001E 00000001 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000EEB62
subq.b  #1,D1              ← D1から1を引く
-t                                                  ↓ ゼロフラグが立った
PC=0010000A USP=000EEB62 SSP=000BBCB4 SR=8004 X:0  N:0  Z:1  V:0  C:0
D  0001001E 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000EEB62
bne.s   $00100006          ← 条件は成立しない
-t
PC=0010000C USP=000EEB62 SSP=000BBCB4 SR=8004 X:0  N:0  Z:1  V:0  C:0
D  0001001E 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000EEB62
addq.l  #2,A7              ← 次の命令を実行
                                Aレジスタへのaddqではフラグが変化しない
-t                                                  ↓
PC=0010000E USP=000EEB64 SSP=000BBCB4 SR=8004 X:0  N:0  Z:1  V:0  C:0
D  0001001E 00000000 00000000 00000000  00000000 00000000 00000000 00000000
A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000EEB64
ori.b   #$00,D0
```

**図8 ブレイクポイントとGコマンド**

```
1) 用意したプログラム
   00100000    moveq   #2,d1
   00100002    bsr.b   $100010        ← サブルーチンの実行
   00100004    dbra    d1,$100002
   00100008    _exit

   00100010    move.w  #'@',-(sp)     ← 画面に@を表示するサブルーチン
   00100014    _putchar
   00100016    addq.l  #2,sp
   00100018    rts

2) ブレークポイントの設定

   -b0 100004

3) Gコマンドによる実行

   -g=100000
   @
          break at 00100004       ← ブレークポイントで止まる
   PC=00100004 USP=000881D4 SSP=000067F2 SR=0000 X:0  N:0  Z:0  V:0  C:0
   D  0001001E 00000002 00000000 00000000  00000000 00000000 00000000 00000000
   A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D4
   dbf     D1,$00100002
   -g              ← 再び実行
   @
          break at 00100004
   PC=00100004 USP=000881D4 SSP=000067F2 SR=0000 X:0  N:0  Z:0  V:0  C:0
   D  0001001E 00000001 00000000 00000000  00000000 00000000 00000000 00000000
   A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D4
   dbf     D1,$00100002
   -g              ← 再び実行
   @
          break at 00100004
   PC=00100004 USP=000881D4 SSP=000067F2 SR=0000 X:0  N:0  Z:0  V:0  C:0
   D  0001001E 00000000 00000000 00000000  00000000 00000000 00000000 00000000
   A  00000000 00000000 00000000 00000000  00000000 00000000 00000000 000881D4
   dbf     D1,$00100002
   -g
   program terminated normally
```

グラムを作成するなら，Sコマンドという有用なコマンドが使用可能である。アセンブラでプログラムを作成するのは難しくない。DB.Xでプログラムを作成できる諸兄なら，なんら問題なく移行できるはずである。図8のプログラムをアセンブラ用のプログラムに直したものを図9-1に挙げておく。

アセンブラ用のプログラムはエディタを使って作成する。福袋ver.2.0をAドライブ，プログラム作成用のディスクをBドライブにセットして作業するのがよかろう。

B>ed test.s

としてエディタを起動し，プログラムを作成されたい。

最初の行は，_exit，_putcharなどのDOSコール名が収めてあるファイルを指定しているところである。これらのDOSコールを使用するには，

DOS _EXIT

のように書かなければならない。そういうルールなのである。続いて，プログラムの入力に入るが，デバッガを使う場合とは異なり，この時点ではサブルーチンのアドレスなどはわからないという点に注意されたい。アドレスは使えないのである。そこで，「loop:」「prnt:」などのラベルを使ってアドレスの代わりとする。

プログラムが作成できたら，「ESC」「E」の順にキーを押してエディタを終了し，図9-2の要領でアセンブル・リンクを行って完成となる。

このプログラムをデバッグするには，

A>db test.x

のようにしてデバッガを起動すればいい。プログラムが自動的に読み込まれて，デバッグ可能な状態になる。Lコマンドでプログラムを表示してブレイクポイントを設定するなり，Gコマンドを使うなり，Tコマンドで追いかけるなり自由にしていただきたい。あとの作業はこれまでの手順と変わらない。

問題のSコマンドを使っているのが図10である。図9でTコマンドを試した諸兄はおわかりかと思うが，Tコマンドはサブルーチンの中まで追いかけていく。これに対してSコマンドは，サブルーチンの中は表示しないで実行し，次の命令に移るのである。ちゃんと動くことがわかっているサブルーチンまで追いかける必要はないというわけである。

## 最後にちょっと悪戯を

最後にちょっとした悪戯心から，図11のようなプログラムを作ってみたので試してみていただきたい。あえて説明はすまい。Tコマンドでひとしきり追いかけたら，

-u 1000

で続きを実行していただけると嬉しい。テキストVRAMの上をプログラムが走っていく様子がご覧いただけるはずである。もともと，テキストVRAMはプログラムを走らせるためのメモリではないので，Gコマンドではうまく動かない。必ず，TコマンドかUコマンドで試していただけるようお願いしておく。

プログラムが走った跡を消去するときには，

-f e40000 e4ffff 0

とすればOKである。

マシン語は吾輩を自由に使いこなすための切符である。吾輩の体の中を自由に触りまくることができ，吾輩の脳に直接命令を与えることができる。こうなったら吾輩はただのデクである。面白くない。そこで，こっちのほうが安全だし簡単だよと高級言語を供給する。

しかしながら，いつの世にも物事をとことんつきつめないと気がすまない輩というのはいるもので（特に理系にはこの類が多い），そういった輩は吾輩が望むと望まないとにかかわらずマシン語の世界へとやってくる。そして，使いづらいと文句をつけては吾輩をけなす。けなして帰っていけばまだいいのだが，より使いやすい環境を生み出そうとして様々なプログラムを作り出してしまう。アセンブラ然り，リンカ然りである。そして，ついにはデバッガまでをも生み出してしまった。

こいつがどんなにマシン語へのアプローチを簡単にしてしまう困ったものかは，いまお伝えしたとおりである。マシン語への特急券といってもよかろう（いや，その特長を考えるなら，マシン語の各停乗車券といったところか)。にもかかわらず，わざわざ吾輩がしゃしゃり出て紹介してしまったのは，吾輩のどこかに極限まで使ってみて，ほしいという欲求が潜んでいるためかもしれない。

マシン語への切符は，しばしば片道乗車券になりがちである。マシン語の世界へ行ったきり帰ってこなくなってしまうのだ。そこまで使い込んでもらえれば，コンピュータとして本望だと考えなければいけないのかもしれない。因果なものだ。

**図9 アセンブラを使ってプログラムを**

```
1) アセンブラ用のプログラムtest.s

        .include doscall.mac    ← _EXITなどが定義してあるファイル

        moveq   #2,d1
loop:   bsr.b   prnt            ← アドレスではなくラベルを使う
        dbra    d1,loop         ← こちらもラベル
        dos     _EXIT

prnt:   move.w  #'@',-(sp)
        dos     _PUTCHAR
        addq.l  #2,sp
        rts

2) アセンブル方法

B>as test
X68k Assembler v2.00 Copyright 1987.88.89.90 SHARP/Hudson
No Fatal error(s)
B>lk test
X68k Linker v2.00 Copyright 1987.88.89.90 SHARP/Hudson
```

**図10 Sコマンドの実際**

```
-t                                  ← トレース
PC=000A9A92 USP=000881D4 SSP=000067F2 SR=8000 X:0  N:0  Z:0  V:0  C:0
D  00000000 00000002 00000000 00000000  00000000 00000000 00000000 00000000
A  000A9990 000A9AA4 00088B64 000733B0  000A9A90 00000000 00000000 000881D4
bsr.s   $000A9A9A
-s                                  ← bsr命令をSコマンドで実行
bsr.s   $000A9A9A

PC=000A9A94 USP=000881D4 SSP=000067F2 SR=0000 X:0  N:0  Z:0  V:0  C:0
D  0001001E 00000002 00000000 00000000  00000000 00000000 00000000 00000000
A  000A9990 000A9AA4 00088B64 000733B0  000A9A90 00000000 00000000 000881D4
dbf     D1,$000A9A92        ← サブルーチンの中は追いかけずに帰ってくる
```

**図11 ちょっと面白いプログラム**

```
00100000  pea      0
00100006  _super
00100008  movea.l  #$e40000,a0
0010000e  move.w   #$3098,(a0)
00100012  jmp      (a0)
```

アドレッシングモード集中講座

# 避けて通れぬ道，アドレッシング

Kageyama Hiroaki 影山 裕昭

MC68000のアセンブラを学習するうえでポイントとなるのが，アドレッシングモードではないだろうか。使いこなせれば強力だが，豊富であるがゆえに最初はとまどうかもしれない。ここではアドレッシングモードだけに絞った解説をお送りする。

BASICでプログラムを組む場合を考えてみると，ある値を定数として扱うのか，変数に取るのか，配列に取るのか，とプログラマはいろいろな形でデータを扱うことになる。

アセンブラでプログラムを組む場合も，変数や配列といったものがレジスタやメモリに変わるだけで，同じようにデータの格納形式をプログラマが指定する。このデータアクセスの指定方法のことをアドレッシングという。

68000には豊富なアドレッシングモードが用意されている（表1）。68000のアドレッシングモードはかなり強力で，複雑なデータ構造でもわりと楽に管理することができる。そのため，プログラマの負担は軽減され，バグの発生も抑えることができる。ということは，開発時間も短縮され，さらにいいプログラムにするための時間が持てることになる。アドレッシングモードの知識があれば，開発効率のいいプログラムを書くことができるのだ。

さて，これからアドレッシングモードを解説していくが，説明は参考書的な要素も多く，ただ字面を眺めていてもつまらないと思う。特に強力なアドレッシングモードについては，サンプルプログラムを紹介するので理解の助けにしてほしい。

表1

| アドレッシングモード | 指定形式 |
|---|---|
| レジスタ直接アドレッシング | ・データレジスタ直接形式<br>・アドレスレジスタ直接形式 |
| アドレスレジスタ間接アドレッシング | ・アドレスレジスタ間接形式<br>・ポストインクリメント<br>・プリデクリメント<br>・ディスプレースメントつき<br>・インデックスつき |
| 絶対アドレッシング | ・絶対ショートアドレス形式<br>・絶対ロングアドレス形式 |
| プログラムカウンタ相対アドレッシング | ・ディスプレースメントつき<br>・インデックスつき |
| イミディエイトデータアドレッシング | ・イミディエイトデータ形式<br>・クイックイミディエイト形式<br>・SR/CCR形式 |

## オペランドサイズについて

アドレッシングモードの説明に入る前に，68000で扱えるオペランドサイズの話をしておく。68000にはオペランドサイズとして，操作対象とするレジスタ幅を8ビット（バイト），16ビット（ワード），32ビット（ロングワード）の3つを指定することができる。指定方法はバイトなら(.b)，ワードなら(.w)，ロングワードなら(.l)を命令に続けて書く。

```
move.w  d0,d1
```

なら，d0レジスタの下位16ビットをd1レジスタの下位16ビットにコピーすることになる。なお，転送先のレジスタの内容は操作対象ビット以外は変更されない。つまり，上の例ならd1レジスタの上位16ビットは影響を受けない。かりに，d0レジスタの下位16ビットが$4321_H$，d1レジスタが$12345678_H$なら，命令実行後のd0レジスタの値は$12344321_H$となる。同様に，

```
move.b  d0,d1
```

とすればd0レジスタの下位8ビットだけがd1レジスタにコピーされ，上位24ビットは影響を受けない。また，

```
move.l  d0,d1
```

ならd0レジスタの全32ビットがd1レジスタにコピーされるので，d1レジスタの全ビットが影響を受ける。

レジスタ間の転送とオペランドサイズの関係を話したが，次にレジスターメモリ間転送との関係についても触れておこう。

いまメモリ内容が，

$10000_H$　$00_H$
$10001_H$　$00_H$
$10002_H$　$00_H$
$10003_H$　$00_H$

として，d1レジスタの値が$12345678_H$としよう。このとき，

```
move.b d1,$10000
```

を実行すれば，メモリ内容は，

$10000_H \leftarrow 78_H$
$10001_H$　$00_H$
$10002_H$　$00_H$
$10003_H$　$00_H$

になる。また，

```
move.w d1,$10000
```

なら，

$10000_H \leftarrow 56_H$
$10001_H \leftarrow 78_H$
$10002_H$　$00_H$
$10003_H$　$00_H$

と，上位8ビットが$10000，下位8ビットが$10001に格納される。16ビットデータであっても，Z80と違い上位バイトと下位バイトが逆に格納されるようなことはない。

さらに，ロングワードの場合は，

$10000_H \leftarrow 12_H$
$10001_H \leftarrow 34_H$
$10002_H \leftarrow 56_H$
$10003_H \leftarrow 78_H$

のようにメモリに転送される。いずれの場合も，転送元のd1レジスタの値は変化しない。

では，オペランドサイズを簡単に説明したところで，本題であるアドレッシングモードの説明に入る。

## レジスタ直接アドレッシング

### ・データレジスタ直接形式

データレジスタとは68000の内部に用意された32ビット長の汎用レジスタで，d0～d7の8つが用意されている。データをしまっておくことのできる箱が8つあると思ってもらえばいい。

このアドレッシングモードはデータレジスタを操作対象にするものをいう。つまり先に出した，

```
move.w  d0,d1
```

もデータレジスタ直接形式である。

ほかにも，

```
add.l  d0,d1
```

muls　d0,d1

など，例を挙げればきりがない。

・アドレスレジスタ直接形式

アドレスレジスタもデータレジスタ同様にA0～A7の8つが用意されている。しかし，A7レジスタはスタックポインタとして使われているので，むやみに値を書き換えたりすることは危険である。また，オペランドサイズにバイトを使うことは許されていない。

このアドレスレジスタを操作対象とするものが，アドレスレジスタ直接形式である。例としては，

movea.l　a0,a1

などであるが，2点ばかり注意が必要である。ひとつは先ほども触れたバイトサイズがサポートされないこと。もうひとつはディスティネーション（転送先）にアドレスレジスタを指定した場合，ソース(転送元)のデータが32ビットに符号拡張（注1）されるということだ。たとえば，

movea.w　d1,a1

なら，ソース側はデータレジスタ直接形式，ディスティネーション側はアドレスレジスタ直接形式である。転送サイズがワードであるから，d1レジスタの下位16ビットを符号拡張した32ビットのデータをa1レジスタにコピーする。であるから，転送サイズがワードであってもa1レジスタの全32ビットが影響を受けることになる。

**注1）** 16ビットで表すことのできる数は，負数を考えなければ0～65535である。負数を扱う場合は最上位ビット（第15ビット）を符号ビットとして扱い，−32768～32767 ($8000_H$～$7FFF_H$) の範囲を表すことができる。32ビットに符号拡張するということは，符号ビットを第16～31ビットにコピーすることである。

d0.w ← $7fff_H$

d1.w ← $8000_H$

を符号拡張すると，

d0.l ← $00007fff_H$

d1.l ← $ffff8000_H$

となる。

## アドレスレジスタ間接アドレッシング

・アドレスレジスタ間接形式

アドレスレジスタの値を操作対象アドレスとする，つまり，アドレスレジスタをポインタとして扱うものである。C言語で変数“adrs”と“＊adrs”の違いのわかる人なら容易に理解できるだろう。しかし，この考え方は初心者にはわかりづらいものであるかもしれない。たとえば，

move.l　(a0),d0

のように，アドレスレジスタをカッコで囲むとアドレスレジスタ間接になる。これはa0レジスタの値を操作対象アドレスとして，そのアドレスから1ロングワードの内容をd0レジスタにコピーすることを命令している。

たとえば，a0＝$10000_H$でメモリ状態が，

$10000_H$　$12_H$
$10001_H$　$34_H$
$10002_H$　$56_H$
$10003_H$　$78_H$

のとき，

move.w　(a0),d0

を実行すると$10000_H$から2バイトの内容がd0レジスタの下位16ビットにコピーされる。その結果，d0レジスタの下位16ビットは$1234_H$になる。もちろん，上位16ビットは影響を受けない。

move.l　a0,d0

のように，アドレスレジスタ直接形式ならa0レジスタの値をd0レジスタにコピーするだけなので，d0＝$10000_H$になる。直接形式と間接形式の違いを理解してほしい。

・ポストインクリメント・アドレスレジスタ間接形式

命令終了後にアドレスレジスタの値を，指定したオペランドサイズに応じて増やすものである。

move.w　d0,(a0)＋

のように，カッコの後ろに“＋”をつけると，ポストインクリメント・アドレスレジスタ間接を指定したことになる。上の命令の動作は，d0レジスタの下位16ビットの値をa0レジスタで示されるアドレスに格納したあと，オペランドサイズがワードであるからa0レジスタの値を2つ増やす。もしオペランドサイズがバイトなら1つ，ロングワードなら4つ，a0レジスタの値を増やす。

具体的な例を挙げよう。d0＝$12345678_H$，a0＝$10000_H$のとき，

move.l　d0,(a0)＋

を実行すると，d0レジスタの全32ビットをa0レジスタの示すアドレスにコピーするので，

$10000_H$ ← $12_H$
$10001_H$ ← $34_H$
$10002_H$ ← $56_H$
$10003_H$ ← $78_H$

になる。このあと，a0レジスタの値はオペランドサイズだけ増えるので，a0＝$10004_H$になる。アドレスレジスタの値は命令終了後に増加することに注意してもらいたい。このアドレッシングモードは，連続するメモリ領域にデータを格納する場合（配列など）に用いると重宝する。

リスト1はブロック転送を行うプログラムである。ポストインクリメント形式を使っているので，非常にすっきりとした形になっている。

・プリデクリメント・アドレスレジスタ間接形式

さっきとは逆に，命令実行前にアドレスレジスタの値をオペランドサイズの分だけ減らすものが，プリデクリメント・アドレスレジスタ間接形式である。これは，

move.w　d0,−(a0)

のように，カッコの前に“−”をつける。a0＝$10002_H$，d0＝$6789ABCD_H$として，メモリの内容が，

$10000_H$　$12_H$
$10001_H$　$34_H$
$10002_H$　$56_H$
$10003_H$　$78_H$

のとき，

move.w　d0,−(a0)

を実行することを考えてみる。まず命令実行前に，アドレスレジスタの値をオペランドサイズ分だけ減らすことをする。オペランドサイズはワードなので2だ。つまり，a0＝$10002_H$－2＝$10000_H$になる。その後，d0レジスタの下位16ビットの値をa0レジスタで示すアドレスにコピーする。結局命令実行後のメモリ内容は，

リスト1

```
 1: *
 2: * ポストインクリメントを使った例
 3: *
 4: * ブロック転送
 5: *
 6: * (a1) → (a2)へd1ワード転送する
 7: *
 8:         .text
 9:         .even
10:
11:         lea.l   data1,a1        * 転送元アドレス
12:         lea.l   data2,a2        * 転送先アドレス
13:         move.w  #100-1,d1       * 転送ワード−1
14: loop:
15:         move.w  (a1)+,(a2)+
16:         dbf     d1,loop
17:
18:         dc.w    $ff00           * DOS _EXIT
19:
20:         .bss
21:
22: data1:
23:         ds.w    100
24: data2:
25:         ds.w    100
26:
27:         .end
```

10000$_H$　AB$_H$
10001$_H$　CD$_H$
10002$_H$　56$_H$
10003$_H$　78$_H$

となり，a0=10000$_H$になる。ポストインクリメント・アドレスレジスタ間接が命令終了後にアドレスレジスタの値を増やすのに対して，プリデクリメント・アドレスレジスタ間接は命令実行前にアドレスレジスタの値を減らすことに注意してもらいたい。このアドレッシングモードを使ったプログラムがリスト2である。DOSコールを使う場合は，必ずといっていいほど使われるアドレッシングモードである。

**・ディスプレースメントつきアドレスレジスタ間接形式**

アドレスレジスタに16ビットの整数（−32768〜32767）のディスプレースメントを（32ビットに符号拡張して）加え，そのアドレスレジスタの値を操作対象アドレスとするものである。たとえば，a0=10000$_H$のとき，

```
move.l  2(a0),d0
```

を実行すると，操作対象アドレスは10000$_H$＋2＝10002$_H$になる。メモリ内容が，

10000$_H$　12$_H$
10001$_H$　34$_H$
10002$_H$　56$_H$
10003$_H$　78$_H$

なら，d0レジスタの下位16ビットは5678$_H$になり，上位16ビットは影響を受けない。

このアドレッシングモードは，1つひとつのデータの大きさが一定でない構造体などをアクセスする場合に非常に便利である。たとえば住所録を作るために，名前に40バイト，住所に80バイト，電話番号に16バイトの構造体を使うとする。このメモリ領域の確保をアセンブラで書くなら，

```
name:  ds.b  40
adrs:  ds.b  80
tel:   ds.b  16
```

となる。先頭の“name”のアドレスを，

```
lea.l  name,a1
```

などとして，a1レジスタに入れておけば，

“name”の先頭アドレスは　0(a1)
“adrs”の先頭アドレスは　40(a1)
“tel”　の先頭アドレスは　120(a1)

で表すことができる。だから“tel”の先頭アドレスをa0レジスタに求めるなら，

```
lea.l  120(a1),a0
```

とすればいい。

**・インデックスつきアドレスレジスタ間接形式**

これはアドレスレジスタの値に，16ビットもしくは32ビットの符号つき整数で表されるインデックスレジスタの値と，8ビットの符号つき整数で表されるディスプレースメントの値を足したものを操作対象アドレスとするという，なんともややこしいアドレッシングモードだ。たとえば，

```
move.w  d0,0(a1,d1.l)
```

のように表す。この場合ディスティネーションがインデックスつきアドレスレジスタ間接形式であり，先頭の0がディスプレースメント，カッコの中のa1がアドレスレジスタ，次のd1.lがインデックスレジスタである。

インデックスレジスタは，アドレスレジスタ，データレジスタの両方を使うことができる。オペランドサイズにバイトは使えず，ワード，ロングワードのいずれかを指定する。省略した場合はワードと解釈される。

例：2(a1,d1) ＝ 2(a1,d1.w)

また，操作対象アドレスの算出はすべてロングワードで行われるため，ディスプレースメントおよびインデックスレジスタにワードを指定した場合は32ビットに符号拡張される。たとえば，a1=10000$_H$，d1=20$_H$のとき，

```
move.b  d1,10(a1,d1.w)
```

を実行すると，d1の下位8ビットのデータがコピーされるアドレスは，

アドレス＝a1＋d1＋10
　　　　＝10000$_H$＋20$_H$＋A$_H$
　　　　＝1002A$_H$

になる。したがって，1002A$_H$に20$_H$が格納される。

このアドレッシングモードは，配列のベースアドレスをa1レジスタ，ベースアドレスからのオフセットをd1レジスタに入れておき，

```
move.w  (a1,d1.w),d0
```

などとすれば，a1レジスタの値を変更することなく，配列の内容をd0レジスタに取り出すことができる。プログラム例としてリスト3を挙げておく。

## アブソリュート(絶対)アドレッシング

**・絶対ショートアドレス形式**

アドレスをワードで与え，それを32ビットに符号拡張して直接指定するものである。そのため指定できるアドレスの範囲が，

00000000$_H$〜00007FFF$_H$
FFFF8000$_H$〜FFFFFFFF$_H$

と限られているので，あまり使われることはないだろう。しかし，メモリ上位に置かれているHuman68kのワークの内容など

リスト2

```
 1: *
 2: * プリデクリメントを使った例
 3: *
 4: * DOSコールを使う
 5: *
 6: * 画面に‘A’を表示する
 7: *
 8:         .text
 9:         .even
10:
11:         move.w  #'A',-(sp)
12:
13:         dc.w    $ff02   * DOS _PUTCHAR
14:         addq.l  #2,sp   * クイックイミディエイト形式
15:
16:         dc.w    $ff00           * DOS _EXIT
17:
18:         .end
```

リスト3

```
 1: *
 2: * インデックス付き
 3: * アドレスレジスタ間接を使った例
 4: *
 5: * d1÷4 の 余りを表示する
 6: *
 7:         .text
 8:         .even
 9:
10:         move.w  #49,d1  * 被除数d1
11:         andi.w  #3,d1   * d1=d1÷4 の余り
12:
13:         lea.l   mes,a1  * メッセージ先頭アドレス
14:         lsl.w   #3,d1   * d1=d1*8
15:
16:         pea.l   (a1,d1.w)
17:
18: * これは
19: *       adda.w  d1,a1
20: *       move.l  a1,-(sp)
21: * と同じだが、a1レジスタの値が変更される。
22: * しかし、a1レジスタを壊してもよければ
23: * クロック数はこっちの方が速い
24:
25:         dc.w    $ff09   * DOS _PRINT
26:         addq.l  #4,sp
27:
28:         dc.w    $ff00   * DOS _EXIT
29:
30:         .data
31: mes:
32:         dc.b    '0だ',13,10,0,0
33:         dc.b    '1だ',13,10,0,0
34:         dc.b    '2だ',13,10,0,0
35:         dc.b    '3だ',13,10,0,0
36:
37:         .end
38:
```

を参照するときは使えるアドレッシングモードかもしれない。アセンブラでは,

```
move.w  $1000.w,d0
```

のように，アドレスの後ろに(.w)をつける。ただし，as.x ver.1.0では絶対ショート形式がサポートされていない。

**・絶対ロングアドレス形式**

操作対象アドレスをロングワードで直接指定する形式である。

```
move.w  d1,$10000
```

は，d1レジスタの下位16ビットの内容を$10000_H$に転送する。その際は最初に話したように，下位16ビットのうち上位8ビットが$10000_H$に転送され，下位8ビットが$10001_H$に転送される。

## プログラムカウンタ(PC)相対アドレッシング

**・ディスプレースメントつきPC相対形式**

プログラムカウンタ（PCレジスタ）と16ビットの符号つき整数のディスプレースメントを足したものを実効アドレスとするものである。

ディスプレースメントは16ビット符号つき整数だから－32768～32767を表せるが，命令の先頭アドレスを基準とすれば－32766～32769になる。なぜかというと，命令を読み込んだ時点でPCレジスタの値が2つ増えているからである。

ディスプレースメントの値はPCレジスタの値と指定アドレスの相対値になるが，as.xなどのアセンブラを使う場合は，ディスプレースメントにラベルを指定しておけば，自動的に両者の相対値を計算してくれる。たとえば，ラベル“data”から1ワードの内容をd0レジスタに取り込むとすると,

```
move.w  data(pc),d0
```

と書くことができる。

このアドレッシングモードはリロケータブルなプログラムを書くときに有用である。リロケータブルとは，どのアドレスにロードしても実効可能なプログラムのことをいい，Human68kでは拡張子がｒのファイルがそれである。しかし，普通の使い方であればｘ形式のファイルであっても，ローディング時にメモリに再配置されるので，特に意識して，リロケータブルに作る必要はない。

もうひとつPC相対形式のおいしいところが，絶対ロングアドレス形式より実行クロックが速いことだ。上の命令も，

```
move.w  data,d0
```

とするより，4クロック速い。ただし，このPC相対形式は,

```
move.w  d0,data(pc)
add.w   d0,data(pc)
```

のように，メモリの内容を書き換えるような場合は使うことができない。

**・インデックスつきPC相対形式**

PCレジスタの値と，16ビットもしくは32ビットの符号つき整数のインデックスレジスタの値と，8ビットの符号つきディスプレースメントの総和を操作対象アドレスとするものである。これは,

```
move.w  label(pc,a0.l), d0
```

のように書く。このアドレッシングモードはリロケータブルなプログラムを書こうというとき以外はあまり使われないだろう。

## イミディエイトデータ・アドレッシング

**・イミディエイトデータ形式**

対象とするデータを直接指定するものである。これは,

```
move.w  #1000,d1
```

のように，必ず'#'に続けてデータを指定する。この命令でd1.wに1000が格納される。また，16進数で指定するなら'#'の後ろに'$'をつけて,

```
move.w  #$640,d1
```

のようにする。いずれの場合もうっかり,

```
move.w  $640,d1
```

と'#'を忘れると，$640番地から1ワードの内容がd1レジスタの下位16ビットにコピーされてしまうので注意すること。

これは特に，Z80のアセンブラから68000に移ってきた人たちにありがちな記述ミスである。また，2進数を扱う場合は'#'の後ろに'%'を書くことになっている。

```
move.b #%0100_0001,d1
```

このように，途中にアンダーバーを入れることも許されている。適当にアンダーバーを挿入することでデータの区切りがよくわかるので，積極的に使ってやろう。

さらに，as.xでは数値を直接指定するだけでなく,

```
move.b  #'A',d1
```

のように，文字を「'」や「"」で囲むことも許されている。この場合，半角文字であればASCIIコード，全角文字であればシフトJISコードの値がd1.wに格納される。上の例ではAを表すASCIIコードである$41_H$が，d1レジスタの下位8ビットに格納される。

**・クイックイミディエイト形式**

これはイミディエイトデータ形式をより高速に行うものである。このアドレッシングモードは,

```
moveq
addq
subq
```

命令でのみ使うことができる。

moveq命令は1バイトのデータをコピーする命令で,

```
moveq  #$80,d0
```

のように書き，イミディエイトデータは符号つき8ビット整数として扱われる。コピーに先立って32ビットに符号拡張されるため，結果はd0レジスタの全ビットが影響を受ける。

レジスタをクリアする命令としてはclr命令があるが，データレジスタの全32ビットをクリアする場合は,

```
moveq  #0,d0
```

などのように，クイックイミディエイト形式を使ったほうが2クロック速いから，知っておいてほしい。

またaddq, subq命令はレジスタのインクリメント，デクリメントを実行するもので，一度に1～8までの範囲で増減させることができる。たとえば,

```
addq.l   #2,a1
subq.w   #8,d2
```

などのようにする。いずれの場合も32ビットに符号拡張して演算が行われる。

**・SR/CCR形式**

SR（ステータス・レジスタ），CCR（コンディション・コード・レジスタ）を操作するものである。詳しくはアセンブラマニュアルを見てもらうことにして，ここでは深く紹介しない。

## 知識だけではダメだ

68000のすべてのアドレッシングモードを駆け足で紹介してきたが，アセンブラを実際に使おうという心構えのない人にとっては非常につまらないものだったかもしれない。

アドレッシングモードというのは，覚えようとして覚えるものではなく，プログラムを書いていくうちに自然に身についてくるものである。

だから，ちょっとでもアセンブラに関心があったら，まずは他人のソースプログラムを真似してプログラムを作ってみることが大事だ。それを繰り返しているうちに知識を使う知恵を身につけ，自分でプログラムが組めるようになるはず。その前段階として，この原稿がこれからアセンブラを学ぼうとする読者諸氏のお役に立つことがあれば幸いである。

とりあえずやってみよう

# 実践アセンブラプログラミング

Hamazaki Masaya 浜崎 正哉

アセンブラにかぎった話ではないが，知識だけを身につけても，実際に自分でいろいろと試してみないと本当の意味での理解はなされない。ここまで知識を身につけた皆さんの，次なる道はプログラミングを実践してみることだ。

いよいよ特集も最後となりました。基本的な概念は，ここまで記事を読み進めてきた皆さんなら理解できたと思います。ここからはプログラミングの過程を中心に説明していきます。そして，どうせなら遊べるサンプルプログラムを作ろう，の精神で少し長めのプログラムを用意しました。

では，お相手はMZ-2200に出会って9年，アセンブラとつきあって8年の私，浜崎がつとめさせていただきます。

## アセンブルのしかた

まず，アセンブルの方法は，

　as /d ［ファイル名］

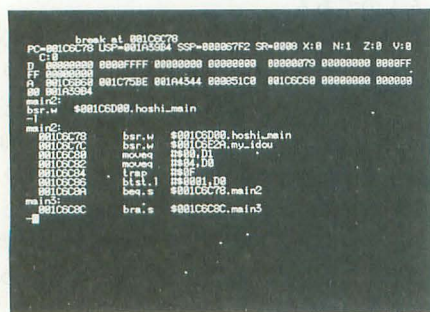
でソースリストがシンボルテーブルつきでアセンブルされることを覚えてしまえば，とりあえず用は足ります。そして，

　LK ［ファイル名］

でリンクという操作を行って".x"の拡張子がついた実行形式のファイルを作ってしまえばアセンブル完了です。

この部分はさらっと流します。わからないことがあったら，マニュアルを参照しましょう。

## プログラミングとは？

どこにでもある教科書的な入門書にあるように，プログラミングに必要な作業は大きく分けて，

1)　設計
2)　コーディング
3)　デバッグ

の3つがあります。まず，何を作るか決めてから，どのようにして使用する言語で記述していくか考えます。最後に，そのプログラムが思ったとおりの動作をするかチェックして，プログラムは完成するのです。

作業の流れは上から下へ一直線に実行していくのが理想でしょう。しかし，現実にそうなることはまずない，ということは経験者なら痛切に感じるはずです。僕がプログラミングをするときには，

1)　全体の構想を軽く考える
2)　必要なサブルーチンの設計
3)　サブルーチンのコーディング
4)　そのサブルーチンをデバッグ

という作業をやっています。2)～3)の作業を中心に完成させる，という具合です。個人レベルでプログラミングを楽しんでいる人たちのほとんどは，このような手順をたどっているのではないでしょうか。

これらの作業のなかで僕が最も気を遣うのは，設計したサブルーチンにバグがあった場合，プログラムがどのような動作をするか？　ということです。これはプログラムが大きくなるにつれて重要になってきます。しっかり把握していないと，どこがどうなっているかわからず，ハマリ状態になってしまいますから注意しましょう。

以前は若さに任せて，このような不都合は力まかせにねじふせていましたが，最近は面倒でも慎重に設計するように心掛けています。設計の部分で苦労しておくと，デバッグが格段に楽ですからね。

また，プログラミングに似合うのは，気力，体力，根気，忍耐など熱血スポ根マンガに出てくるような言葉だと思います。バグが出たときなど，コーディングのやり直しや，設計自体をやり直さなくてはならないような事態がいくらでも発生します。そんなときに必要なものが上記のクサイ言葉たちです。

## 星は飛んでくぞっと

一般論はこれぐらいにして，何を作りましょうか。と，考えている横でA氏がAMIGAのスクリーンセイバーを実行させて遊んでいるのが目につきました。SIONの背景にあるような星が中心から放射状に広がっていくヤツです。じぃ～っとながめているうちにアルゴリズムを考えだし，頭の中でコーディングまで始めてしまいました。そうして，これぐらいだったら簡単に実現できるだろう，という結論に達して……。

よし，サンプルプログラムは星が流れるデモにしよう。しかし，ながめているだけではものたりないので，SIONのようにキー操作で星の動きを変化させるようにします。

おおまかにどんなルーチンが必要か，簡単に考えると，

1)　星を出現させる
2)　星を動かす
3)　星を表示させる
4)　キー入力で星の座標を変化させる

以上があれば大丈夫でしょう。星の表示はスプライトで行い，遠近感らしきものを出すため大きさの違う星を飛ばすことにします。ちょっとおおまかすぎる気がしますが，最初はこれぐらいで考えて，徐々に煮詰めていけばいいのです。

## 星の動きをあばく

まず，星を放射状に広がって飛ばすアルゴリズムはどうすればいいのでしょうか。最初に僕が考えたアルゴリズムは，星の出現座標を画面の中心にし，乱数によってX，Y方向の移動量を設定，星の移動は初めに求めた移動量を加算していく，というものでした。しかし，よく考えてみると星の移動量がひとつひとつ違い，てんでばらばらの動きをするし，星が中心から出現するのも不自然な感じがします。

こりゃいかんと思い，考え直したアルゴリズムは，星を表示している座標をあるていどシフトした値を移動量とするものです（図1）。これだったら，星を画面のどこに出現させても放射状に飛んでくれるし，速度の違う星を出現させたいときには，シフトする値を変化させるだけですみます。

出現したあと移動しっぱなしでは困るので，移動カウンタを用意してカウンタが0になったら出現位置に戻すようにします。出現位置は初めの1回だけ求めてワークに格納しておきます。カウンタが0になったとき，出現位置をX,Y座標にセットするだけですむので，初期化ルーチンを呼ぶ必要がなく，少し高速になるでしょう。

また，スプライトの表示にはIOCSコールのSP_REGSTを使うことにします。

## キー入力は?

さて，キー入力によって星を動かすルーチンです。これは何も悩む必要なく簡単に実現できます。キー入力されたら移動量を増加させて，その移動量をそれぞれの星のX,Y座標に加算してやればいいのです。ポイントはキー入力がなされたら，ある一定値を星の座標に加算していくのではなく，増加させた移動量を加算するところです。

そして，これらの処理にキー入力がなくなったら自動的に減速させる，という処理を加えると慣性がきいているような動きをさせることができます。移動量の増減範囲が大きければ大きいほど，ゆったりとした動きになります。

## ワークエリアの構造

ここで，ワークエリアの構造とワークエリアにどのようにアクセスするかを説明します。星1個に必要なワークエリアは，

1）表示するスプライトナンバー。0以外の値だとワークに動かす星が存在する，というフラグも兼ねています
2）表示するスプライトキャラクタナンバー。速度の違う星のフラグも兼ねています
3）X座標
4）Y座標
5）カウンタ
6）出現X座標
7）出現Y座標

のようになります。それぞれワードサイズの大きさなので，合計14バイトが1個の星に必要なワークエリアサイズです。全体では7ワード×出現させる星の数の分だけ領域を確保しておけばいいことになります。

こうして確保したワークエリアは図2のように並んでいると考えます。で，アセンブラではどのようにして特定番号のワークをアクセスするのでしょうか。領域を確保したといっても，アセンブラ側できちんと区切りをつけてくれるわけでもなく，ワークはメモリ上にどばーっと並んでいるだけです。したがってワークエリア管理というものは，プログラマが自分でしなくてはなりません。このへんはアセンブラを使い始めた人にとって，引っ掛かる点でしょう。

基本的にはワークエリア先頭アドレスを基準にして考えていきます。先ほど述べたように，1個分のワークエリアサイズは14バイトで，ひとつのワークが2バイトになっています。したがって，n番目の星のX座標を変えたいときのワークアドレスは，

hoshi_work＋n＊14＋4

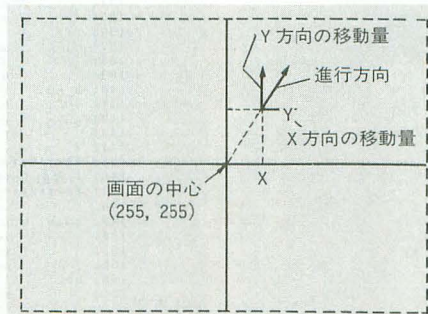
と求めることができます。

ところが，アクセスするたびにこんな計算をしていたのでは，面倒臭くてしょうがありません。こういうときには間接アドレッシング＋ループを用いて処理してやるのが普通です。ここで星を移動させるためのメインルーチン（hoshi_main）を見てください。最初にワークエリアの先頭アドレスをa1レジスタにセットし，出現させる星の個数だけループを繰り返しているのがわかるでしょう。

ループの中では，まずワークにデータが存在しているかチェックします。ない場合はa1レジスタに14を足して次のブロックの先頭アドレスを計算を行っているところにジャンプ，データがあれば移動メインルーチン（hoshi_move）に制御を移します。

ほかのサブルーチンでも同様の処理をしていますので探してみてください。

## 流れはどうなっている?

具体的にどんなサブルーチンが必要なのかつかめたでしょうか。今度はそれらのサブルーチンを組み合わせて，全体の流れが円滑に運ぶようにします。プログラムを見ながら説明を読んでいってください。

まず，メインルーチンは，

1）画面モードの初期設定
プログラム起動時の画面モードを保存して，スプライトが使えるように画面モードを512×512モードにします
2）ワークエリアの初期設定
星のワークエリアのクリア。スプライト面の初期化。星のPCGデータを定義。星のデータを初期化しています。ここで星のデータを初期化するためにhoshi_app, hoshi_moveをコールしているのはなぜだと思います？　理由は教えてあげません。リストを見てわからなかったら，試しに98行を注釈行にしてアセンブル，実行してみてください。ま，そういうことです。
3）星の移動ルーチンをコール
4）キー入力によって移動量をセットするルーチン（my_idou）をコール
5）ESCキーが押されなかったら3)に制御を戻す
6）終了処理
画面モードの復帰。キーバッファのクリアをしてからプログラムを終了します

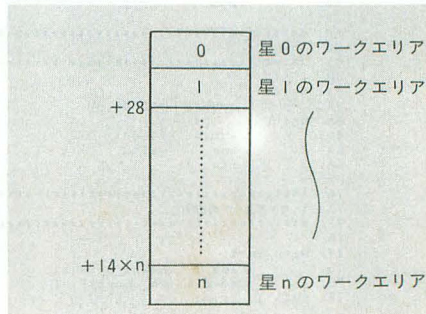
のようになっています。

次にそれぞれのサブルーチンはどうなっているか，どのようにつながっているか見ていきます。

・星の出現ルーチン（hoshi_app）
ワークに空きエリアがあるかチェックして，空きエリアが存在していればフラグのセット，X,Y座標のセット，カウンタの初期化をします。

・星の移動ルーチン（hoshi_move）
カウンタを−1して移動が終了したかチェック。終了していなければ，X,Y座標にキー入力で発生した移動量を座標に足す。終了していればカウンタの初期化，X,Y座標に出現座標をセットする。あとは両方とも座標から移動量の計算（hdx_cal）をして，新しくセットされた座標で星の表示(hoshi_

図1

図2

put）をします。
・X,Y座標から増分を決定して座標に加算（hdx_cal）

星の大きさによってシフトする値を決定。座標軸を（255, 255）にして移動量を計算し座標に加算します。

・星を表示（hoshi_put）

まず，X,Y座標が画面内に収まっているかチェックします。収まっていない場合には，プライオリティを0にしてスプライトレジスタの書き込みをしています。

・キー入力によって移動量を決定するメインルーチン（my_idou）

キー入力ルーチン（key_in）をコールして，d2レジスタにX方向の移動量，d3レジスタにY方向の移動量を得ます。それをもとに移動量の計算，移動量が範囲内に収まっているかチェック。もしも，移動量が0であったら移動量の減速（x_gensoku,y_gensoku）を行います。

・キー入力ルーチン（key_in）

IOCSコールを使って，直接キーボードからデータを読みます。そしてそれに従って，X,Y方向の移動量をd2, d3レジスタに格納します。

・移動量が範囲内に収まっているかチェック（idou_chk）

タイトルのとおりのことをします。

・移動量の減速（x_gensoku,y_gensoku）

移動量が0になるように移動量を増減していきます。

＊　　＊　　＊

以上ですべてのサブルーチンを解説してきました。初心者の人たちは，解説がどのようにアセンブリ言語で記述されているか確かめながら読み進めるといいでしょう。

何かわからないことがあったら今月号の記事を参照するなり，マニュアルを読むなりして，なんとか理解していきましょう。理解できたらスピード調節機能などをつけたり，ひとつある不都合のデバッグにチャレンジしてみてください。

不都合というのは，画面の中心部に出現した星が張りついてしまうことです。さあ，どこを直せばいいのかな？

さて，自己流ですがプログラミングについていろいろ書いてきました。こういったものの考え方は，丸暗記だけでは身につきません。何度も失敗を繰り返し，自分で考えた経験がものをいいます。とおり一辺倒の使い方なんて面白くもなんともないでしょう。別にきれいなプログラムスタイルにこだわらなくてもいいですから，自分なりのプログラミングを楽しんでください。

### リスト

```
  1: *
  2: *  星を動かすプログラム
  3: *
  4:
  5:         .include        iocscall.mac
  6:         .include        doscall.mac
  7:
  8: h_count         equ     119     *動かす星の総数
  9: data_size       equ     14      *星1つあたりのワークサイズ
 10:
 11: flag_1          equ     0       *スプライトナンバー
 12: flag_2          equ     2       *スプライトキャラクター
 13: h_x             equ     4       *X座標
 14: h_y             equ     6       *Y座標
 15: h_cnt           equ     8       *移動カウンタ
 16: app_x           equ     10      *出現X座標
 17: app_y           equ     12      *出現Y座標
 18: idoumax         equ     63      *移動量の上限
 19: counter         equ     100
 20:
 21: FPACK   macro   callno
 22:         dc.w    callno
 23:         endm
 24:
 25: __RAND          equ     $fe0e
 26:
 27:         .text
 28:         .even
 29:
 30: main:
 31:         move.l  #-1,d1
 32:         IOCS    _CRTMOD
 33:         move.l  d0,scr_mod              *画面モード保存
 34:         moveq.l #0,d1
 35:         IOCS    _CRTMOD                 *画面モードを512*512にする
 36:         IOCS    _B_CUROFF
 37:
 38:         bsr     work_init
 39: main2:
 40:         bsr     hoshi_main
 41:         bsr     my_idou
 42: *       bsr     wait
 43:
 44:         moveq.l #0,d1
 45:         IOCS    _BITSNS                 *ESCキーチェック
 46:         btst    #1,d0
 47:         beq     main2
 48:
 49:         move.l  scr_mod,d1
 50:         IOCS    _CRTMOD                 *画面モード復帰
 51: ret:
 52:         IOCS    _B_CURON
 53:         move.w  #$ff,-(sp)
 54:         move.w  #6,-(sp)
 55:         DOS     _KFLUSH
 56:         addq.l  #4,sp
 57:         DOS     _EXIT
 58:
 59: ************************************
 60: * ウェイト
 61: ************************************
 62:
 63: wait:
 64:         move.l  #$2000,d0
 65: wait2:
 66:         sub.w   #1,d0
 67:         bne     wait2
 68:         rts
 69:
 70: ************************************
 71: * ワークエリアの初期化
 72: ************************************
 73:
 74: work_init:
 75:         lea.l   hoshi_work,a1
 76:         move.w  #h_count*7,d1
 77: wk2:
 78:         clr.w   (a1)                    *星のワークエリアをクリア
 79:         dbf     d1,wk2
 80:
 81:         IOCS    _SP_INIT                *スプライトの初期化
 82:         IOCS    _SP_ON                  *スプライト面の表示
 83:
 84:         moveq.l #0,d1                   *パターンコード
 85:         moveq.l #0,d2                   *パターンサイズ
 86:         lea.l   hoshi_pat,a1            *パターンアドレス
 87:         IOCS    _SP_DEFCG               *星のパターン定義
 88:
 89:         moveq.l #4,d1                   *パターンコード
 90:         moveq.l #0,d2                   *パターンサイズ
 91:         lea.l   hoshi2_pat,a1           *パターンアドレス
 92:         IOCS    _SP_DEFCG               *星のパターン定義
 93:
 94:         move.w  #h_count,d1             *星のデータを初期化
 95: wk3:
 96:         movem.l d1,-(sp)
 97:         bsr     hoshi_app
 98:         bsr     hoshi_main
 99:         movem.l (sp)+,d1
100:         dbf     d1,wk3
101:         rts
102:
103: ************************************
104: *  星をスクロールさせる
105: ************************************
106:
107: hoshi_main:
108:         lea.l   hoshi_work,a1
109:         move.w  #h_count,d6
110: hm2:
111:         tst.w   flag_1(a1)
112:         beq     hm3
113:         bsr     hoshi_move
114: hm3:
115:         adda.l  #data_size,a1
116:         dbf     d6,hm2
117:         rts
118:
119: ****************************************
120: *  星の移動メインルーチン
121: ****************************************
122:
123: hoshi_move:
124:         subq.w  #1,h_cnt(a1)    *  カウンター1
125:         bne     h_m3
126:                                 *星を再び出現させる
127: hoshi_reset:
128:         move.w  #counter,h_cnt(a1) *カウンタ初期化
129:         move.w  app_x(a1),h_x(a1)  *X座標リセット
130:         move.w  app_y(a1),h_y(a1)  *Y座標リセット
131:         bra     h_m2
132: h_m3:
133:         move.w  dx,d1                   *X座標に移動増分を足す
134:         asr.w   #3,d1
135:         sub.w   d1,h_x(a1)
136:
137:         move.w  dy,d1                   *Y座標に移動増分を足す
138:         asr.w   #3,d1
139:         sub.w   d1,h_y(a1)
140: h_m2:
141:         bsr     hdx_cal
142:         bsr     hoshi_put
143:         rts
144:
145: ************************************
146: *  星を出現させる
147: ************************************
148:
149: hoshi_app:
150:         lea.l   hoshi_work,a1
151:         moveq.l #1,d4                   *スプライトナンバー
152:         move.w  #h_count,d6
153: hs2:
154:         movem.l d4-d6/a1,-(sp)
```

```
155:           tst.w    flag_1(a1)     *ワークエリアが空いている?
156:           bne      hs3
157:           bsr      hoshi_set
158:           bsr      hoshi_main
159: hs3:
160:           movem.l  (sp)+,d4-d6/a1
161:           adda.l   #data_size,a1
162:           addq.w   #1,d4
163:           dbf      d6,hs2
164:           rts
165:
166: ************************************
167: *  出現させる星の座標をセット
168: ************************************
169:
170: hoshi_set:
171:           move.w   d4,flag_1(a1)  *フラグ(スプライトナンバー)セット
172:           move.w   #counter,h_cnt(a1)     *カウンタ初期化
173:
174:           FPACK    __RAND
175:           and.w    #$01ff,d0
176:           move.w   d0,h_x(a1)       *X座標セット
177:           move.w   d0,app_x(a1)     *出現X座標セット
178:
179:           FPACK    __RAND
180:           and.w    #$01ff,d0
181:           move.w   d0,h_y(a1)       *Y座標セット
182:           move.w   d0,app_y(a1)     *出現Y座標セット
183:
184:           moveq.l  #1,d1            *星の明るさを決定
185:           FPACK    __RAND
186:           and.w    #1,d0
187:           beq      ho_s2
188:           moveq.l  #0,d1
189: ho_s2:
190:           move.w   d1,flag_2(a1)
191:           rts
192:
193: ************************************
194: *  X,Y座標から増分を決定して加算
195: ************************************
196:
197: hdx_cal:
198:           moveq.l  #5,d1
199:           tst.w    flag_2(a1)       *星の明るさによって移動量を変える
200:           beq      hd_2
201:           moveq.l  #6,d1
202: hd_2:
203:           move.w   h_x(a1),d0
204:           sub.w    #255,d0          *座標を255中心に
205:           asr.w    d1,d0
206:           add.w    d0,h_x(a1)
207:
208:           move.w   h_y(a1),d0
209:           sub.w    #255,d0          *座標を255中心に
210:           asr.w    d1,d0
211:           add.w    d0,h_y(a1)
212:           rts
213:
214: ************************************
215: *  星を表示
216: ************************************
217:
218: hoshi_put:
219:           movem.l  d1-d5,-(sp)
220:
221:           clr.l    d2
222:           clr.l    d3
223:           moveq.l  #3,d5            *プライオリティ
224:           move.l   #$8000_0000,d1
225:           add.w    flag_1(a1),d1    *スプライト番号
226:
227:           move.w   h_x(a1),d2
228:           bmi      hp_no            *X座標範囲チェック
229:           cmp.w    #552,d2
230:           blt      hp_ok
231: hp_no:
232:           moveq.l  #0,d5            *プライオリティ0(消去)
233: hp_ok:
234:           move.w   h_y(a1),d3
235:           bmi      hp_no2           *Y座標範囲チェック
236:           cmp.w    #552,d3
237:           blt      hp_ok2
238: hp_no2:
239:           moveq.l  #0,d5            *プライオリティ0(消去)
240: hp_ok2:
241:           move.l   #$01_00,d4       *PCG番号
242:           move.w   flag_2(a1),d0
243:           add.w    d0,d4
244:           IOCS     _SP_REGST        *スプライトレジスタ設定
245:           movem.l  (sp)+,d1-d5
246:           rts
247:
248: ************************************
249: *  キー入力によって移動量を設定
250: ************************************
251:
252: my_idou:
253:           bsr      key_in
254:           tst.w    d2
255:           bne      my_i2
256:           bsr      x_gensoku        *X方向の減速
257:           bra      my_i3
258: my_i2:
259:           add.w    dx,d2            *X方向の移動量を加算
260:           bsr      idou_chk         *X方向の移動量をチェック
261:           move.w   d2,dx            *X方向の移動量をセット
262: my_i3:
263:           tst.w    d3
264:           bne      my_i4
265:           bsr      y_gensoku        *Y方向の減速
266:           rts
267: my_i4:
268:           move.w   d3,d2
269:           add.w    dy,d2            *Y方向の移動量を加算
270:           bsr      idou_chk         *Y方向の移動量をチェック
271:           move.w   d2,dy            *Y方向の移動量をセット
272:           rts
273:
274: ********************************************
275: *  入力されたキーによって移動量を増減
276: ********************************************
277:
278: key_in:
279:           moveq.l  #0,d2                  *X方向の移動量
280:           moveq.l  #0,d3                  *Y方向の移動量
281:                                           *4キーのチェック
282:           move.l   #8,d1
283:           IOCS     _BITSNS
284:           btst     #7,d0
285:           beq      up_chk
286:           subq.w   #1,d2
287: up_chk:                                   *8キーのチェック
288:           btst     #4,d0
289:           beq      right_chk
290:           subq.w   #1,d3
291: right_chk:                                *6キーのチェック
292:           move.l   #9,d1
293:           IOCS     _BITSNS
294:           btst     #1,d0
295:           beq      down_chk
296:           addq.w   #1,d2
297: down_chk:                                 *2キーのチェック
298:           btst     #4,d0
299:           beq      chk_out
300:           addq.w   #1,d3
301: chk_out:
302:           rts
303:
304: ********************************************
305: *  移動量が範囲内に収まっているかチェック
306: ********************************************
307:
308: idou_chk:
309:           cmp.w    #-idoumax,d2
310:           bge      chk_d2
311:           move.w   #-idoumax,d2
312:           bra      chk_d3
313: chk_d2:
314:           cmp.w    #idoumax,d2
315:           ble      chk_d3
316:           move.w   #idoumax,d2
317: chk_d3:
318:           rts
319:
320: ********************************************
321: *  X,Y方向の移動量を減速
322: ********************************************
323:
324: x_gensoku:
325:           tst.w    dx                     *移動量=0?
326:           bne      x_g2
327:           rts
328: x_g2:
329:           moveq.l  #1,d1
330:           move.w   dx,d0
331:           bmi      x_g3
332:           move.w   #-1,d1
333: x_g3:
334:           add.w    d1,dx
335:           rts
336:
337: y_gensoku:
338:           tst.w    dy                     *移動量=0?
339:           bne      y_g2
340:           rts
341: y_g2:
342:           moveq.l  #1,d1
343:           move.w   dy,d0
344:           bmi      y_g3
345:           move.w   #-1,d1
346: y_g3:
347:           add.w    d1,dy
348:           rts
349:
350: ************************************
351: *  ワークエリア
352: ************************************
353:
354: scr_mod:
355:           dc.l     00                     *画面モードを保存
356: dx:
357:           dc.w     00                     *X方向の移動増分
358: dy:
359:           dc.w     00                     *Y方向の移動増分
360:
361: ************************************
362: *  星のパターン
363: ************************************
364:
365: hoshi_pat:
366:           dc.l     $00000000
367:           dc.l     $00000000
368:           dc.l     $00000000
369:           dc.l     $000ff000
370:           dc.l     $000ff000
371:           dc.l     $00000000
372:           dc.l     $00000000
373:           dc.l     $00000000
374: hoshi2_pat:
375:           dc.l     $00000000
376:           dc.l     $00000000
377:           dc.l     $00000000
378:           dc.l     $000e0000
379:           dc.l     $00000000
380:           dc.l     $00000000
381:           dc.l     $00000000
382:           dc.l     $00000000
383:
384: ********************************************
385: *  星のワークエリア
386: *    .w  フラグ(スプライトナンバー)
387: *    .w  フラグ2(スプライトキャラクターナンバー)
388: *    .w  X座標
389: *    .w  Y座標
390: *    .w  カウンタ
391: *    .w  出現X座標
392: *    .w  出現Y座標
393: ********************************************
394:
395: hoshi_work:
396:           ds.w     7*120
397:           .end
```

# まず音階，そして和声の基礎

Taki Yasushi 瀧 康史

打ち込んだデータを演奏させるだけがコンピュータミュージックではありません。もっと創造的な道だってあります。この連載では最終的にコンピュータを使って作曲/編曲を行うことを目標とします。まずは理論編。がんばってついてきてください。

先日，ファルコムのゲームミュージックよりアレンジされたアルバム，“Pre・Primer”を購入しました。一応，ドラゴンスレイヤーIVのBGM「城・街」（1曲目の原曲）と，ピラミッドソーサリアンのBGM「Click! Click!」（4曲目の原曲）以外の曲は，一度ならずとも原曲は聞いたことがあるはずなのですが（しばらく流し聞きした限りでは），4曲目のソーサリアンオープニングのアレンジ以外の曲は，サッパリ馴染みのメロディに気がつきませんでした。

かといって，このアルバムがアレンジが突飛で変なアルバムではなく，小気味のよいきれいなメロディの心地よいアルバムで，「俗にいういい曲ばかり集めた」という宣伝文句はだてではないな。こんなふうに，アレンジができたらいいだろうなという感覚に浸ることができました。

最近，X68000にMIDIが普及しつつあって，パソコン通信などをしていると，なにかのゲームミュージックのアレンジというものをよく見かけます。音源がグレードアップして，お気に入りのゲームミュージックを自分なりにアレンジしたいというのは，コンピュータミュージックが普及してきた現在となっては，音楽好きの人には当たり前のことかもしれません。

アレンジされたBGMを聞きながら思うのですが，センスがあるのにオクターバー（完全8度の音かぶせ，もう1オクターブ下の音などを重ねる奏法）しかアレンジの方法を知らず，もうちょっと，アンサンブルを勉強していればいいのに，ということをしばしば感じるのです。

コンピュータを使って音楽を行っていく際には，アレンジという問題が常につきまといます。今回から始まる連載では，音楽のアレンジや作曲といった作業に欠かせない考え方と基本テクニックを修得することを目的として，音楽の基礎理論から実践までを解説してきます。当面は「いかにしてうまくアレンジするか」という点について考えてみたいと思います。

先ほどの，“Pre・Primer”のようには美しくアレンジできなくとも，コードの仕組みやそのほかの音の基本的な構造を覚えるだけで，ずいぶんアレンジが違ってきます。「私は，センスだけでミュージックをアレンジするのだ」というのは，見当違いですよ。数学だってなんだって，先人たちの偉大な知識を無視しては，決してそのことはきわめられないでしょう。そのことは音楽でも同じことがいえるのです。

## Multipleって？ Chordって？

皆さんがよく知っているFM音源にはMultipleというパラメータというものが存在します。FM音源を使ったことのある人なら，Multipleは基準音の整数倍の周波数音を指定するものだということは周知のことかと思います。

ただ，そもそもなぜ，C1,G1,C2,E2,G2,Bb2,C3（数字はオクターブの差）と，倍音は連なっていくのでしょうか？ よく，シンセサイザのカタログにある非倍音を作り出すことができるとか，OPMのDT2を使うことによって，非倍音を発音することができるといわれます。そこまで意識される倍音というのは何者なのでしょうか？

コードの本にはよく，ピアノのペダルを踏んでおいて，低音Cを鳴らすと共鳴という現象が起きてほかの音が鳴るのを聞き取ることができる……と書いてあります。その音が倍音で，ずっとずっと無限に続きます。このへんのことは，高校で物理を専攻し共鳴を習えば想像がつくと思いますので説明はパスします（長くなりますので）。

ピアノが近くにあって，せっかちの人は，すでにやってるかもしれませんが，ちょっと待ってくださいな。ちなみに，人間の耳に聞こえるのは，個人差はあってもだいたいG2ぐらいまでです。わかりましたか？ 倍音とは共鳴によって自然に発せられる音（周波数）だったのです。

このなかにある音階は，C,それからE,Gですね。これらを協和音といいます。この倍音3つを寄せ集めると，もっとも基本的なCを根音としたメジャーコード（長三和音）となるわけです。このメジャーコードは，倍音列の協和音のなかで組み立てられた和音なので，自然和音と考えてもよく，和音の基礎となるものです。

まずメジャーコードの構成について解説しておきましょう。ここで出てくる「完全何度」とかいう数え方は，とりあえずは1991年3月号のOh!X中野氏の記事の46ページか音楽の教科書を引っ張り出して見てください。一応，なんでそうなるか？ ということは，このあとスケール（音階）を覚えるときに，説明を入れますので，そのときよく覚えてください。さあ，図1を見て，ここはひとまずこういうもんだという感じで，意味を読み取ってください。なぁに，簡単なもんですよ。リラックスして見てくださいな。

それでは，C,E,Gが協和音だということは話しましたよねぇ？ この自然和音の各音間の音程（相対差）を調べてみることにします。ぐるぐる回したり，1オクターブ

図1 度数

を上に上げたりずらしたり。そうすると，(図2およびリスト1)C1E1G1ではC1,G1のあいだが，完全5度，C1,E1,のあいだが長3度,E1,G1のあいだが短3度。3月号の記事では，Cに対して書いてありますから，相対的に置き換えてください。

なんとなく感覚的にはわかるでしょ？　それから1回転して，E1,G1,C2にすると，E1,C2間が短6度,G1,C2のあいだが,完全4度，2回転してG1，C2，E2にすると，G1,E2のあいだが長6度となります。

全部あわせると，完全5度，完全4度，長3度，短3度，長6度，短6度の，6組の協和音程でのみ構成されています。不協和音程は使用されてませんね？　この数え方は相対なので注意してください。

頭で考えるよりも，楽器が近くにある人は試してみたほうがいいと思います。指で数えてなるほどな，とうなずいてください。楽器のない人は，SOUND PRO-68Kや，そのほかの鍵盤の出るツールで試したほうがいいでしょう。協和音となる音程は，ある音を基準として考えてくださいね。

ここでひとつ，とりあえず覚えていてください。協和音程だけで構成されたコード(和音)を協和音と，協和音程と不協和音程によって構成されたコードを不協和音といいます。協和音程というのは，さっきの6つと，完全1度，完全8度をあわせた8つです。そのほかは不協和音程です。

図3を見ながら考えてみます。仮に基準音をCとすると，完全1度が，同じ音C，完全8度は1オクターブ上のC。完全5度はG……となるわけです。

リスト2に，それぞれを鳴らすものを掲載するので，響きと雰囲気を味わってみてください。一応鍵盤を基準とした図を書いておきます(図4)。図を見てしっかり考えてください。こんなものは理屈ですから。

### リスト1

```
10 m_init()
20 for i=1 to 3
30 m_alloc(i,100):m_assign(i,i)
40 next
50 m_trk(1,"c1 e1 g1")
60 m_trk(2,"e1 g1<c1")
70 m_trk(3,"g1<c1 e1")
80 m_play()
```

### リスト2

```
10 m_init()
20 for i=1 to 2
30 m_alloc(i,100):m_assign(i,i)
40 next
50 m_trk(1,"                1")
60 m_trk(3,"          g1<c1 e1")
70 m_play()
```

## Scale(音階)とTriad Chord(三和音)

音階は，昔はたくさんありました。いま，よく使われてるのはそのなかでも2つだけ，イオニアモード（長音階）と，エオリアモード（短音階）と呼ばれるものだけです。それぞれ，CDEFGABとABCDEFGですが，この2つ，どのあたりに違いがあるかというと，半音上がる位置が違うのです。同じようにCで始まっても，CDE♭FG♭A♭B♭Cになると，ABCDEFGAと音階がそっくりに聞こえてしまいます。

またまた，鍵盤を思い出してください。イオニアモード（長調）では，EとF，BとCのあいだに黒鍵はありません（半音しか上がってないということ)。音階はそのまま一定の値で上げていくと，1オクターブは

図2　和音の回転

図3　基準音Cの協和音程と各種三和音

図4

黒鍵を含めた12段階となります。ところが白鍵のみでは8つ。その代わり，CとDのあいだにはひとつの黒鍵が入っていて，同じようにいくつも黒鍵があるわけですね。ですから，イオニアモードではCとDのあいだは半音が2つ，すなわち全音上がっているのに対し，EとF,BとCのあいだでは半音しか上がっていません。

これに対して，エオリアモード(短調)は，AとB，EとFで半音です。

昔は，Bからあがるスケールや，Fからあがるスケールなどが，それぞれありました。いまは，それが2つ。イオニアモードはメジャースケール，エオリアモードはマイナースケールとなっています。

もっとも最近では，このなんとかモードってのは使われませんけどね。短音階（マイナースケール）は実際は，本当に暗いイメージになってしまうため，それを避けるために和声的短音階や旋律的短音階という，いくぶん明るくするための特殊なスケールがあります（メジャースケールというのは，小学校のとき音楽でならった長調のことで，マイナースケールというのは短調のことです)。これらについては，オブリガッド(裏に礎するメロディみたいなもの）によるアレンジのとき取り組んでみます。

さあ，これだけで脳味噌はウニに早変わりです！

さてさて，このメジャースケールとマイナースケールというのは，さっきもちょっといっていたように，なにもメジャーはCから，マイナーはAからと決まっているわけではありません。

さっきの2つは，それぞれC（Maj:シーメジャーと読む，Majは普通省略）それからAm（エーマイナー）といって，これはそのままコードの名前ですがここではスケールを指していることに注意してください。これら2つは黒鍵をまったく押さえない調律です。楽譜を見るとわかるのですが，いちばん初め(楽譜のいちばん左端)には，#(シャープ）も♭（フラット）もありません。これがさっきいったCmになると，半音上がる位置をあわせるために，♭が3つつきます。逆にA（Maj）だと，#は3つです。

いや，間違えないでくださいよ。マイナーだから♭だとか，メジャーだから#ってことはありません。現にEmは#はひとつです（ちなみにロック，ポピュラー系の楽譜ではEmで書いてあるものがなぜか多いです。本当はEbmとかが多いんですけど）。しかし，やっぱり，なぜか♭のつく曲はフラットな曲が多いし,#のつく曲はシャープです

**図5　音階いろいろ**

よね？（なぜかってわけでもないんですけどね)。黒鍵を含めた11音を基準として，それぞれメジャースケール，マイナースケールが存在します。

C(Maj) はハ長調。Amはイ短調です。そうすると，種類的には全部で22個あるわけですが，(音階はC,C#, D,D#, E……Bと，あわせて11個ありますのでメジャーとマイナーで22個ってわけです）音階の持つイメージや雰囲気はメジャー（長調）とマイナー（短調）で２つ。

あとはちょうど，カラオケなどでいう「キー」の違いというものです。11段階あるわけですね。だから，A(Maj)のスケールの曲よりC(Maj)のほうが，A,A#, B,Cと短３度（半音にして３つ上）高いわけです。それぞれ22個の図を記しておきます(図５)。音程の違いはぜひプログラムにして確認してみてください。どうです？ キーの違いの雰囲気はわかりましたか？

ところでここでスケールを話したのはなぜかというと，このスケールというものは，コードと密接に関わりあいがあるからです。ここでこのマイナースケール，メジャースケールの各音の上に三和音を組み立ててみます。三和音というのは，ある音の上に長３度，短３度のいずれかの音を重ねて，その上にさらに長３度，短３度のいずれかの音を重ねた和音です。

Cを基準音として三和音を組み立てると（三和音は省略)，長短，短長，長長，短短と，図３の下のように2×2＝４組できますね？ 長短がメジャーコード。短長がマイナーコード。長長がオーギュメントコード。短短がディミニッシュコードと呼ぶことを，とりあえず覚えていてください。

そのなかで，長短とできた三和音，すなわち，メジャー（長三和音）とマイナー（短三和音）ですがこれらは協和音となります。また，短短，長長でできた三和音，すなわちディミニッシュコード（減三和音）オーギュメントコード（増三和音）ですが，これらは不協和音です。とりあえずここではそういうものだということを覚えていてください。

スケールの話に戻りましょう。それぞれのスケール上ではオクターブのなかに７音あります。この音を順に最初の音から，１度，２度，……７度と呼び，これを，I,II,III, IV,V,VI,VIIのローマ数字で表してみることにします。

ここでそれぞれの上に三和音を作ります。仮にもC (Maj) なら白鍵をそれぞれひとつ飛ばしに，Amでも白鍵をそれぞれひとつ飛ばしに叩いたものです。

そしてその和音が長和音なら大文字（I IV V)，短和音なら小文字（i iv v)，減和音なら(ii° vii°），増和音なら(III＋)のように書き表すと，

長音階C(Maj)の上にできる三和音は，

I,ii,iii,IV,V,vi,vii°, I

コードでは，

C,Dm,Em ,F ,G,Am,Bdim,C
(T)　　　(S)(D)

短音階Ａｍの上にできる三和音は，

i,ii°, III,iv, v,VI,VII,i

コードでは，

Am,Bdim, C ,Dm,Em, F,G ,Am
(T)　　　(S) (D)

となります。

それぞれ，音階の１度上の和音を主和音（トニックコード)，４度上の和音を下属和音(サブドミナントコード)，５度上の和音を属和音（ドミナントコード）と呼びます。これらは，主要三和音（スリーコード）と呼ばれ，楽曲が作成される基礎となるもっとも重要なコードです。上の表中では，(T) (S) (D) のついているコードです（ここでも取りあえずこの時点では重要だということだけ覚えていてください)。

それからこのスリーコードに対して，もうひとつ気がつかなくてはならないことがあります。それは，長音階（メジャースケール）では，主要和音はすべて「長和音（メジャーコード)」となり，短音階（マイナースケール）では，主要和音はすべて「短三和音（マイナーコード)」となっていることです。ここに，短音階が暗いイメージをみごとにかもしだし，長音階が明るいイメージを盛り上げる秘訣があるわけです。

＊　＊　＊

とりあえず，今回はこのあたりで終わりにしておきます。覚え初めは結構ややこしもので，すぐ間違えてしまうものですがだいたいが把握できていれば，肝心なときには出てくるものだと思いますよ。昔，ヤマハの音楽教室で，ド～ミ～ソ～と始まって～と，コード進行をひそかに習った覚えがあるのですが，あのときはそれを歌うことがとても楽しくて，ピアノの先生の前にこぞって並んだ記憶があります

思い出話になってしまいましたが，とりあえず今回はこれだけではつまらなさすぎるので，ゲームミュージックを１曲アレンジしてみました。

曲は，X68000ユーザーならだれでも知っていそうな曲をやりたかったのですが，私，実は１年ちょっと前まで，PC-9801ユーザーでありまして，昔のX68000をよく知らないのです。それで，今回はどうしようもないのですが，イースがX68000に移植されましたので，イースのオープニング曲「Feena」を奏でてみたいと思います。

ただ，私がこの曲をアレンジしたときにはX68000版イースを見たことがなかったので，それとはまったく関係がないアレンジとなってしまっています（それよりできが悪くても，怒らないでくださいね)。

神秘的なこの曲はイースのヒロインFeenaにぴったりで私はとても気にいってます。元にしたのはPC-9801版イースの「Feena」です。FM３声＋SSG３声から８声へのアレンジです。

今回は大変忙しくて，急いで作ったものなので，あまり「オカズ」を入れることができなくてすみません（簡単なアレンジですが)。ぜひ，なんとかして聞いてみてください。

Rolandのミュージくんの宣伝ではないんですけど，私も「音楽が嫌いな人はいない」という言葉を，少なからず信じています。４歳のときから音楽という魔力に取りつかれてしまった私ですが，いまから読者たちあなた方を，私と同じように音楽の魔力に引き入れて見せたいと思っています。

これから何回か連載しますが，自己紹介を含めて，今回のあと書きをこれで終わりにしたいと思います。

それでは，また，来月号でお会いしましょう。

**図6　C(MaJ)，Amのスケールと三和音**

## 図7 Feena

Feena
By Kohju
©日本ファルコム
♩=60
Tenuto
16va
mp
Vel=100
CH=1 AcouBass 1
8va
fff
Vel=80
f
Elec Org 4
Vel=40
CH=1
p
Elec Org 1
CH=2
Em
Bm
AcouPiano1
CH=3
C#m7-5
F#m

## 内蔵音源のための“Feena”8声による変奏

ご存じ日本ファルコムのRPGイースよりオープニング曲“Feena”です。この楽譜はMUSIC PRO-68K[MIDI]上で入力，8声にアレンジして出力したものです。

MUSIC PRO-68Kをお持ちの方はこれを入力して演奏してみるもよし，楽譜から音符だけを拾ってMMLに直すのもよいでしょう。ただ，音取りやアレンジの参考にする場合はすでにこの楽譜自体がアレンジされているものだということに注意してください。

このデータはもともとRolandのシンセサイザD70用に作成されていました。だいたいの音色名は記入されていますが，具体的な音色については特に指定してありません。音声の数はできる限り，8声にとどめてありますので，できる人はこれにあった音色を作成してFM音源に再アレンジしてみてください。

イントロ部分の2ブロックを除いて，あとは一貫してピアノが入っていますが，すべて手で弾けるものにしてあります。ただ，Bメロにあたっては，2重奏になっているので（こんなものをもし弾くとしたら）演奏者が2人必要です。また，Bメロの6度のアルペジオ（G6）のところですけど，あれは耳で聞いてちょっと違和感あるでしょ。ちなみにわざとです。

# GSフォーマットを斬る

Tama Tamaki たま たまき

話題の新音源SC-55。今回は音源標準化を目指してRolandがSC-55に採用したGSフォーマットの詳細を解説します。あわせて標準音色とドラムキットも紹介。デスクトップミュージックの夜明けは近い……かもしれない。

最近，DTMの世界では“音源の標準化”という言葉がささやかれています。シーケンサのデータフォーマットならともかく，楽器という商品は“個性”がウリなだけに「どこまで浸透するか？」という懸念があるのも事実。そうした状況のなか，昨年末に発表され，今年5月に発売されたGSフォーマット準拠のSC-55。はたして市場はGSフォーマットを受け入れてくれるのでしょうか？

## なぜ標準化が必要か？

現在のシンセサイザ（以下，音源）はそれぞれ独自のプリセット音色を持っているため互換性はありません。いままでは，

　プリセット音色＝シンセサイザの顔

という認識があったので，音色配列を統一するという考えや，統一する必要性もありませんでした。

しかし，ここ1，2年のDTMブーム(？)により，音楽データを制作するアマチュアの人たちが急激に増加し，またパソコン通信を介して不特定多数の人々とデータ交換をするといったことも当たり前となりつつあります。

ところが，いまのところ音楽データや音源の互換性がないので，送る側，受け取る側双方とも同じ音源を持っていなければ，正常に演奏させることができないのです。

ここで誤解されると困るのであえて説明しますが，音楽データについて互換性がないといっているのは音源固有の情報が含まれているということであり，ソフトウェア的なデータフォーマットの互換性のことではありません。

現在，シーケンサのデータフォーマットについてはスタンダードMIDIファイルという形式で統一しようという動きがあり，最近のシーケンサや音楽ソフトはたいていスタンダードMIDIファイルを採用しているか，またはコンバータが付属しています。それがなくても，ミュージくん/郎のSNGファイルにコンバートできるようになっているか，読み込みだけはできるように配慮されています。このようにデータフォーマットそのものはなんとかなるのですが，音源というのはハードウェアなのでユーザー側では対処できないのです。

たとえば，音源Aで演奏することを前提として作成した音楽データを音源Bで演奏させると，音色配列が異なるため違った音で演奏されます。プログラムチェンジ情報を類似した音色に差し替えたとしてもベロシティなどのハードウェアに起因するダイナミクス情報が異れば，正常に演奏することは困難です。このような問題をクリアするには，

1)　演奏する音源用にリアレンジする
2)　機種依存しない音楽データを作成する
3)　音色配列および音源制御について統一規格を制定する

ことが必要です。

1)については，センスと労力でクリアできるでしょう。

2)については，固有の音源を用いることにより得られた音楽的表現力が失われますし，そのような音楽データを作成するにはかなりの知識が必要であり，誰でもできるわけではありません。

3)は楽器メーカー間での問題であり，ユーザー側では対処できません。しかし，音色配列の互換性，および音源制御の統一規格が求められているのも事実です。

このような市場背景の下，DTMのパイオニア的存在であるRolandが提案したものがGSフォーマットです。

なお，いまのところはRoland独自のものであり，MIDI協議会が規格として定めたものではありません。今後，MIDI規格に盛り込まれるか否かは定かではありません。念のため。

## どういう規格なのか？

GSフォーマットは前述した2つの問題，つまり音色配列と音源制御について規定したものです。

従来の音源のプリセット音色は128種類以下しかありませんでした。これはMIDIのプログラムチェンジで切り替えられる数の上限です。ユーザー定義音色についてもこのなかに含まれます。

この点に関してはメーカー側も結構苦労したみたいで，たとえばYAMAHAの例を見てみると，プログラムチェンジのうち実際に音色に割り当てるのは1から64ないし100くらいで，残りの部分にバンクセレクト的機能を割り当てることにより，128の壁を乗り越えています。

しかし，このような方法で各メーカー独自の方法で対処するのは，規格統一に反する行為であり，異なるメーカー間の音源を用いて演奏させるときにさまざまなトラブルを発生しかねません。これではMIDI規格の意味が失われます。

そこで，MIDI規格協議会は「音色配列のバンクセレクトは，コントロールチェンジ0番（上位）と32番（下位）で行う」というふうに追加規定しました。これで，音色のバンクセレクトについてはMIDI規格として統一されたわけです。

これにより，従来128種類しか切り替えできなかった音色は，最大で16384バンク×128音色をひとつの音源で扱えるようになりました。

しかし，音色配列についてはまだMIDI規格としては規定されていません。そこで，まずRolandは独自の規格としてGSフォーマットを考案したわけです。

しかし，音色配列を統一しても，ベロシティなどのダイナミクス情報が異なれば演奏表現上の互換性はとれません。また，現

在MT-32などで行われているようなエクスクルーシブメッセージによる演奏表現（たとえば演奏中のリバーブの切り替えや，マスターボリュームによるフェードアウトなど）はその音源固有のものであり，互換性はありません。たとえ同じLA方式のD-110においてもです。

そこで，GSフォーマットでは従来エクスクルーシブメッセージで行われていた演奏表現上必要と思われる機能をコントロールチェンジ情報に追加しています。

## GSフォーマットの概要

それではGSフォーマットについて，ポイント別に説明していきます。

### トーンマップ

GSフォーマットのもっとも基本的かつ重要なポイントがトーンマップです。このトーンマップがなければ，従来の音源とたいして変わりはありません。

トーンマップは128個の音色を1バンクとし，合計で128バンクあります。GSフォーマットに準拠した音源であれば，まずコントロールチェンジ0番でバンクを指定し，プログラムチェンジで音色を指定します。バンクセレクトの下位（B0 20 nn）は使用されていません。

たとえば，MIDIチャンネル2の音色をバンク8の31番のFeedback Gt.に変更したい場合は，

B1 00 08 C1 1E（hex）

というようにします。

GSフォーマットではバンクのことをバリエーションと呼んでいます。また，バンク番号によって以下のような特殊な呼び方がされており，重要な意味を持っています。

**●キャピタルトーン（以下，メインキャピタル）**

バリエーション番号0の音色のことで，Rolandによると基本的な楽器音はほとんど含んでいるということです。メインキャピタルはGSフォーマットに対応したすべての音源において完全に保証されています。また，7種類までバリエーションを持つことが可能となっています。Humanのディレクトリ構造にたとえていうならば，ルートディレクトリ的存在です。

**●サブキャピタルトーン**

バリエーション番号が8,16,24,32,40,48,56の音色のことで，キャピタルトーンに対する本質的な意味でのバリエーションです。これらの音色は似通った音色ですが，異なる楽器名を持っています。8の倍数はサブキャピタルと覚えましょう。

サブキャピタルも7種類までバリエーションを持つことが可能です。Humanのディレクトリ構造にたとえていうならば，サブディレクトリ的存在です。

その他のバリエーション番号の音色群のことを単にバリエーションと呼んでいます。Humanのディレクトリ構造にたとえたなら，さらに下の階層ディレクトリというところでしょう。

### トーンマップの階層構造

GSフォーマットのトーンマップは図2のような階層構造を持っています。それでは，この階層構造を利用した代替音色ロジックについて，SC-55を例にして説明しましょう。

例1：

バリエーション10を選択しておき（Bn 00 0A），プログラムチェンジ9番に切り替える（Cn 08）としましょう。まず，GSフォーマットの音源はバンク10に音色が登録されているかどうかを参照します。SC-55では，割り当てられていないので，バリエ

図1　GSのメインキャピタル（括弧内はSC-55の使用パーシャル数）

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ピアノ | 1 (1) Piano 1 | 2 (1) Piano 2 | 3 (1) Piano 3 | 4 (2) Honky-Tonk Piano | 5 (1) E. Piano 1 | 6 (1) E. Piano 2 | 7 (1) Harpsichord | 8 (1) Clav. |
| クロマチック・パーカッション | 9 (1) Celesta | 10 (1) Glockenspiel | 11 (1) Music Box | 12 (1) Vibraphone | 13 (1) Marimba | 14 (1) Xylophone | 15 (1) Tubular-bell | 16 (1) Santur |
| オルガン | 17 (1) Organ 1 | 18 (1) Organ 2 | 19 (1) Organ 3 | 20 (1) Church Org. 1 | 21 (1) Reed Organ | 22 (2) Accordion Fr | 23 (1) Harmonica | 24 (2) Bandneon |
| ギター | 25 (1) Nylon-str. Gt | 26 (1) Steel-Str. Gt | 27 (1) Jazz Gt. | 28 (1) Clean Gt. | 29 (1) Muted Gt. | 30 (1) Overdrive Gt | 31 (1) DistortionGt | 32 (1) Gt. Harmonics |
| ベース | 33 (1) Acoustic Bs. | 34 (1) Fingered Bs. | 35 (1) Picked Bs. | 36 (1) Fretless Bs. | 37 (1) Slap Bs. 1 | 38 (1) Slap Bs. 2 | 39 (1) Synth Bass 1 | 40 (1) Synth Bass 2 |
| ストリングス&オーケストラ | 41 (1) Violin | 42 (1) Viola | 43 (1) Cello | 44 (1) Contrabass | 45 (1) Tremolo Str | 46 (1) PizzicatoStr | 47 (1) Harp | 48 (1) Timpani |
| アンサンブル | 49 (1) Strings | 50 (1) Slow Strings | 51 (1) Syn. Strings1 | 52 (2) Syn. Strings2 | 53 (1) Choir Aahs | 54 (1) Voice Oohs | 55 (1) SynVox | 56 (2) OrchestraHit |
| ブラス | 57 (1) Trumpet | 58 (1) Trombone | 59 (1) Tuba | 60 (1) MutedTrumpet | 61 (2) French Horn | 62 (1) Brass 1 | 63 (2) Synth Brass1 | 64 (2) Synth Brass2 |
| リード | 65 (1) Soprano sax | 66 (1) Alto sax | 67 (1) Tenor sax | 68 (1) Baritone sax | 69 (1) Oboe | 70 (1) English Horn | 71 (1) Bassoon | 72 (1) Clarinet |
| パイプ | 73 (1) Picccolo | 74 (1) Flute | 75 (1) Recorder | 76 (1) Pan flute | 77 (2) Bottle Blow | 78 (2) Shakuhachi | 79 (1) Whistle | 80 (1) Ocarina |
| シンセ・リード | 81 (2) Square Wave | 82 (2) Saw Wave | 83 (2) Syn. Calliope | 84 (2) Chiffer Lead | 85 (2) Charang | 86 (2) Solo Vox | 87 (2) 5th Saw Wave | 88 (2) Bass&Lead |
| シンセ・パッドなど | 89 (2) Fantasia | 90 (1) Warm Pad | 91 (2) Polysynth | 92 (1) Space Voice | 93 (2) Bowed Glass | 94 (2) Metal Pad | 95 (2) Halo Pad | 96 (1) Sweep Pad |
| シンセSFX | 97 (2) Ice Rain | 98 (2) Soundtrack | 99 (2) Crystal | 100 (2) Atmosphere | 101 (2) Brightness | 102 (2) Goblin | 103 (1) Echo Drops | 104 (2) Star Theme |
| エスニック | 105 (1) Sitar | 106 (1) Banjo | 107 (1) Shamisen | 108 (1) Koto | 109 (1) Kalimba | 110 (1) Bag Pipe | 111 (1) Fiddle | 112 (1) Shanai |
| パーカッシブ | 113 (1) Tinkle Bell | 114 (1) Agogo | 115 (1) Steel Drums | 116 (1) Woodblock | 117 (1) Taiko | 118 (1) Melo Tom 1 | 119 (1) Synth Drum | 120 (2) Reverse Cym. |
| SFX | 121 (1) Gt. GretNoise | 122 (1) Gl. Kkeyclick | 123 (1) Seashore | 124 (2) Bird | 125 (1) Telephone 1 | 126 (1) Helicopter | 127 (2) Applause | 128 (1) Gun Shot |

ーション10のサブキャピタルであるバンク8を参照します。ところが，SC-55ではここにも音色が割り当てられていないので，メインキャピタルであるバンク０のCelestaが選択されます。

例２：

先ほど，バリエーション10を選択しておいたので，今度はプログラムチェンジ５番(Cn 04)に切り替えてみます。例１のロジックに従って，まずバンク10を参照します。SC-55ではここに音色は割り当てられていないので，サブキャピタルであるバンク８を参照します。今度はDetuned EP 1という音色が割り当てられているので，これが選択されます。

このようにGSフォーマット準拠の音源は指定された音色がないと代替音色で演奏してしまうのです。ただし，この方法でも再生する音源によって音質感が変わってしまうことは事実ですし，それはしかたのないことなのであきらめましょう。また，将来，違う音源チップで作られたGSフォーマットの音源においても，たとえ同一バンクの同一プログラム番号の音であっても，音質感は変わります。誤解のないように。

また，バンク64以降については，ユーザーエリアやスペシャルといったGSフォーマットの規格外の用途に使用されるため，このエリアに割り当てられている音色は代替音色処理を行いません。たとえば，SC-55のバリエーション127のMT-32セットはGSフォーマットには含まれていないので，代替音色は割り当てられません。

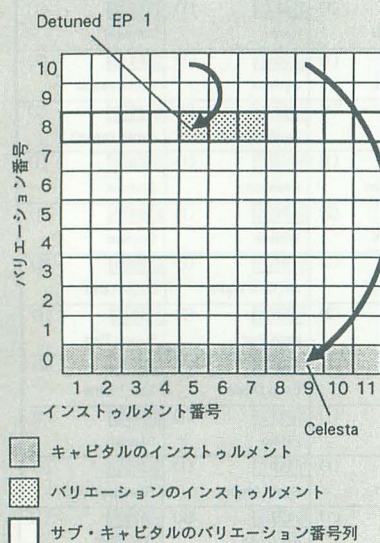
なお，プログラムチェンジ120～127の音色はSFXに割り当てられています。SFXに代替音色を割り当てることは効果上，ほとんど無意味なので階層構造は持っていません。GSフォーマットでは一度バンクセレクトを送ると，次のバンクセレクトが送られてくるまで有効です。また，バンクセレクトはプログラムチェンジが送られてくるまで処理が保留されます。たとえば，SC-55で該当するMIDIチャンネルを出力するトラックの先頭（セットアップのために１小節空けておくと非常に便利）で次のように送信しておくと，わざわざ手作業でMTモードにしなくても，MT-32の曲を演奏させることができます。

```
C1 00 7F    ;CH2のTr.
C2 00 7F    ;CH3のTr.
C3 00 7F    ;CH4のTr.
C4 00 7F    ;CH5のTr.
C5 00 7F    ;CH6のTr.
C6 00 7F    ;CH7のTr.
C7 00 7F    ;CH8のTr.
C8 00 7F    ;CH9のTr.
;ここまではMT-32のバンクを選択
B9 7F       ;CH10のTr.
;これはDRUM SETの切り替え
```

図2

## ドラムセットの入れ替え

ドラムセットの入れ替えができるのもGSフォーマットの特長のひとつといえます。要するにドラムパートに割り当てられているチャンネルでプログラムチェンジを行うと，ドラムセットが切り替わるのです。

たとえば，スタンダードセット（プログラムチェンジ１番）を使用しているときにパワーセット(インストゥルメント番号17)に切り替えるということができます。

キーアサインについても定義されており，同系列の音色は同じキーに割り当てられます。これにより，ドラムセットを切り替えても正常に演奏できます。

ドラムパートにもユーザーエリアやスペシャルといったエリアがありこれはGSフォーマットの規格外の用途に使用されます。たとえばSC-55のバリエーション127のCMドラムセットはGSフォーマットには含まれていません。

ちなみに，ドラムパートはバンクセレク

図3　ドラムキット例(SC-55)

| ノート・ナンバー | 1:Standard Set<br>33:Jazz Set | 9:Room Set | 17:Power Set | 25:Electronic Set | 26:TR-808 Set | 41:Brush Set | 49:Orchestra Set |
|---|---|---|---|---|---|---|---|
| 27 | High Q | | | | | | Closed Hi-Hat [EXC1] |
| 28 | Slap | | | | | | Pedal Hi-Hat [EXC1] |
| 29 | Scratch Push | | | | | | Open Hi-Hat [EXC1] |
| 30 | Scratch Pull | | | | | | Hide Cymbal |
| 31 | Sticks | | | | | | |
| 32 | Square Click | | | | | | |
| 33 | Metronome Click | | | | | | |
| 34 | Metronome Bell | | | | | | |
| 35 | Kick Drum 2 | | | | | | Consert BD2 |
| 36 | Kick Drum 1 | | MONDO Kick | Elec BD | 808 Bass Drum | | Concert BD1 |
| 37 | Side Stick | | | | 808 Rim Shot | | |
| 38 | Snare Drum 1 | | Gated SD | Elec SD | 808 Snare Drum | Brush Tap | Concert SD |
| 39 | Hand Clap | | | | | Brush Slap | Castanets |
| 40 | Snare Drum 2 | | | Gated SD | | Brush Swirl | Concert SD |
| 41 | Low Tom 2 | Room Low Tom 2 | Room Low Tom 2 | Elec Low Tom 2 | 808 Low Tom 2 | | Timpani F |
| 42 | Closed Hi – hat [EXC1] | | | | 808 CHH [EXC1] | | Timpani F# |
| 43 | Low Tom 1 | Room Low Tom 1 | Room Low Tom 1 | Elec Low Tom 1 | 808 Low Tom 1 | | Timpani G |
| 44 | Pedal Hi – hat [EXC1] | | | | 808 CHH [EXC1] | | Timpani G# |
| 45 | Mid Tom 2 | Room Mid Tom 2 | Room Mid Tom 2 | Elec Mid Tom 2 | 808 Mid Tom 2 | | Timpani A |
| 46 | Open Hi – hat [EXC1] | | | | 808 OHH [EXC1] | | Timpani A# |
| 47 | Mid Tom 1 | Room Mid Tom 1 | Room Mid Tom 1 | Elec Mid Tom 1 | 808 Mid Tom 1 | | Timpani B |
| 48 | High Tom 2 | Room Hi Tom 2 | Room Hi Tom 2 | Elec Hi Tom 2 | 808 Hi Tom 2 | | Timpani c |
| 49 | Crash Cymbal 1 | | | | 808 Cymbal | | Timpani c# |
| 50 | High Tom 1 | Room Hi Tom 1 | Room Hi Tom 1 | Elec Hi Tom 1 | 808 Hi Tom 1 | | Timpani d |
| 51 | Ride Cymbal 1 | | | | | | Timpani d# |
| 52 | Chinese Cymbal | | | Reverse Cymbal | | | Timpani e |
| 53 | Ride Bell | | | | | | Timpani f |
| 54 | Tambourine | | | | | | |
| 55 | Splash Cymbal | | | | | | |
| 56 | Cowbell | | | | 808 Cowbell | | |
| 57 | Crash Cymbal 2 | | | | | | Concert Cymbal2 |
| 58 | Vibra – slap | | | | | | |
| 59 | Ride Cymbal 2 | | | | | | Concert Cymbal1 |
| 60 | High Bongo | | | | | | |
| 61 | Low Bongo | | | | | | |
| 62 | Mute High Conga | | | | 808 High Conga | | |
| 63 | Open High Conga | | | | 808 Mid Conga | | |
| 64 | Low Conga | | | | 808 Low Conga | | |
| 65 | High Timbale | | | | | | |
| 66 | Low Timbale | | | | | | |
| 67 | High Agogo | | | | | | |
| 68 | Low Agogo | | | | | | |
| 69 | Cabasa | | | | | | |
| 70 | Maracas | | | | 808 Maracas | | |
| 71 | Short Hi Whistle [EXC2] | | | | | | |
| 72 | Long Low Whistle [EXC2] | | | | | | |
| 73 | Short Guiro [EXC3] | | | | | | |
| 74 | Long Guiro [EXC3] | | | | | | |
| 75 | Claves | | | | 808 Claves | | |
| 76 | High Wood Block | | | | | | |
| 77 | Low Wood Block | | | | | | |
| 78 | Mute Cuica [EXC4] | | | | | | |
| 79 | Open Cuica [EXC4] | | | | | | |
| 80 | Mute Triangle [EXC5] | | | | | | |
| 81 | Open Triangle [EXC5] | | | | | | |
| 82 | Shaker | | | | | | |
| 83 | Jingle Bell | | | | | | |
| 84 | Belltree | | | | | | |
| 85 | Castanets | | | | | | |
| 86 | Mute Surdo [EXC6] | | | | | | |
| 87 | Open Surdo [EXC6] | | | | | | |
| 88 | | | ----- | | | | Applause |

トを無視します。

## GSフォーマットの最低音源スペック

GSフォーマットの音源は，

1) 最大発音数24ボイス以上

2) 16マルチティンバーであること

と定義されています。

また，ダイナミックボイスアロケーション（DVA）機能およびパート間優先順位，ボイス（パーシャル）リザーブ機能によって発音制御をすることになっています。DVAとは，ある1パートで発音中に別の音を鳴らさなければならなくなった場合，空きチャンネルをみつけて同時に発音させる機能です。

つまり，SC-55の音源仕様がGSフォーマットの最低スペックというわけです。

## コントロールチェンジによる音源制御

前に述べたとおり，GSフォーマット準拠の音源では，演奏表現上必要と思われる音源制御をエクスクルーシブを用いないで，すべてコントロールチェンジに割り振っています。

従来，エクスクルーシブが使用される代表的な例として，MT-32のリバーブコントロールが挙げれらますが，GSフォーマット準拠の音源であればコントロールチェンジでリバーブのセンド量を変更できます。

ポルタメント関係やソフトといった演奏効果もコントロールチェンジで行えます。

マスターチューニング関連やピッチベンドセンシビリティはコントロールチェンジのRPN（Registerd Parameter Number：MIDI規格で機能が規定されている拡張領域）を使用することにより設定を変更できます。

そういえば，昔，よくキーボーディストが演奏しながらリアルタイムでシンセサイザーのつまみをいじっていたあの技（音色パラメータの変更）みたいなこともGSの規格内でできますが，あれは結構機種依存性があります。

GSフォーマットでは，機種依存しそうなパラメータはコントロールチェンジのNRPN（Non Registerd Parameter Number）という，いわば機器固有の拡張領域に割り振っています。

## エクスクルーシブメッセージも定義

GSフォーマット準拠の音源はその音源のモデルID（SC-55は45H）のほかにGSフォーマットのモデルID（42H）を持っています。そして，このモデルIDで送られてくるエクスクルーシブメッセージを受信します。

GSフォーマットで受信するエクスクルーシブメッセージは，エフェクタ，キーシフト，マスターチューニング，マスターボリューム，パート情報，音色パラメータなどです。

筆者が思うに音色パラメータまで定義されているのはちょっと不思議な気がしますが，とりあえずTVF/TVA（周波数/音量変化）関係なので，Rolandの音源であればよほどのことがない限り変更されることはないと思います。しかし，多分，いまのところは暫定仕様で，将来変更されることがあるかもしれません。

## GSフォーマットの拡張性，そして

GSフォーマットに準拠して作成された

表1

MIDI SOUND GENERATOR　　Date : Jan. 24 1991

Model SC-55　　MIDI インプリメンテーション・チャート　　Version : 1.00

| ファンクション… | | 送信 | 受信 | 備考 |
|---|---|---|---|---|
| ベーシック チャンネル | 電源 ON 時 | × | 1 - 16 | |
| | 設定可能範囲 | × | 1 - 16 each | 記憶可能 |
| モード | 電源 ON 時 | × | モード 3 | |
| | メッセージ | × | モード 3, 4 (m = 1) | ＊2 |
| | 代用 | ＊＊＊＊＊＊＊＊＊ | | |
| ノート ナンバー | | × | 0 - 127 | |
| | 音域 | ＊＊＊＊＊＊＊＊＊ | 0 - 127 | |
| ベロシティー | ノート・オン | × | ○ | |
| | ノート・オフ | × | × | |
| アフタータッチ | キー別 | × | ＊1 | |
| | チャンネル別 | × | ＊1 | |
| ピッチ・ベンダー | | × | ＊1 | 分解能：12ビット |
| コントロールチェンジ | 0, 32 | × | ＊3 MSB only | バンク・セレクト |
| | 1 | × | ＊1 | モジュレーション |
| | 5 | × | ＊3 | ポルタメント・タイム |
| | 6, 38 | × | ＊3 | データ・エントリー |
| | 7 | × | ＊1 | ボリューム |
| | 10 | × | ＊1 | パンポット |
| | 11 | × | ＊1 | エクスプレッション |
| | 64 | × | ＊1 | ホールド1 |
| | 65 | × | ＊1 | ポルタメント |
| | 66 | × | ＊1 | ソステヌート |
| | 67 | × | ＊1 | ソフト |
| | 91 | × | ＊3 (Reverb) | 汎用エフェクト1 |
| | 93 | × | ＊3 (Chorus) | 汎用エフェクト3 |
| | 98, 99 | × | ＊1 | NRPN LSB, MSB |
| | 100, 101 | × | ＊1 | RPN LSB, MSB |
| | 120 | × | ○ | オール・サウンド・オフ |
| | 121 | × | ○ | リセット・オール・コントローラー |
| プログラムチェンジ | | × | ＊1 | |
| | 設定可能範囲 | ＊＊＊＊＊＊＊＊＊ | 0 - 127 | |
| エクスクルーシブ | | ○ | ○ | |
| コモン | ソング・ポジション | × | × | |
| | ソング・セレクト | × | × | |
| | チューン | × | × | |
| リアルタイム | クロック | × | × | |
| | コマンド | × | × | |
| その他 | ローカル ON/OFF | × | × | |
| | オール・ノート・オフ | × | ○ (123 - 127) | |
| | アクティブ・センシング | ○ | ○ | |
| | リセット | × | × | |
| 備考 | | ＊1 ○×切り換え可能<br>＊2 m！= 1の場合もm = 1として扱う<br>＊3 コントロール・チェンジすべての○×切り換えによってのみ○×切り換え可能 | | |

モード 1：オムニ・オン，ポリ　　モード 2：オムニ・オン，モノ　　○：あり

モード 3：オムニ・オフ，ポリ　　モード 4：オムニ・オフ，モノ　　×：なし

表2　GS音源のNRPN設定

| MSB | LSB | MSB | |
|---|---|---|---|
| 01H | 08H | mmH | ビブラート・レイト（相対変化）<br>mm:0EH-40H-72H（-50-0-+50） |
| 01H | 09H | mmH | ビブラート・デプス（相対変化）<br>mm:0EH-40H-72H（-50-0-+50） |
| 01H | 0AH | mmH | ビブラート・ディレイ（相対変化）<br>mm:0EH-40H-72H（-50-0-+50） |
| 01H | 20H | mmH | TVF カットオフ・フリクエンシー（相対変化）<br>mm:0EH-40H-72H（-50-0-+50） |
| 01H | 21H | mmH | TVF レゾナンス<br>mm:0EH-40H-72H（-50-0-+50） |
| 01H | 63H | mmH | TVF&TVA エンベロープ・アタック・タイム（相対変化）<br>mm:0EH-40H-72H（-50-0-+50） |
| 01H | 63H | mmH | TVF&TVA エンベロープ・デッケイ・タイム（相対変化）<br>mm:0EH-40H-72H（-50-0-+50） |
| 01H | 66H | mmH | TVF&TVA エンベロープ・リリース・タイム（相対変化）<br>mm:0EH-40H-72H（-50-0-+50） |
| 18H | rrH | mmH | ドラム・インストゥルメント・ピッチ・コース（相対変化）<br>rr:ドラム・インストゥルメントのキー番号<br>mm:00H-40H-7FH（-64-0-+63 半音） |
| 1AH | rrH | mmH | ドラム・インストゥルメントTVAレベル（絶対変化）<br>rrH:ドラム・インストゥルメントのキー番号<br>mm:00H-7FH（0-最大） |
| 1CH | rrH | mmH | ドラム・インストゥルメント・パンポッド（絶対変化）<br>rr:ドラム・インストゥルメントのキー番号<br>mm:00H, 01H-40H-7FH（ランダム，左-中央-右） |
| 1DH | rrH | mmH | ドラム・インストゥルメント・リバーブ・センド・レベル（絶対変化）<br>rr:ドラム・インストゥルメントのキー番号<br>mm:00H-7FH（0-最大） |

音楽データなどは，GSフォーマットに準拠した音源であれば，モデルの新旧に関らず，データ修正をすることなしに，演奏できることが保証されます。

これはエンドユーザーが購入した音源の陳腐化を防ぐとともに，データ資産の継承も意味します。

身近な例を述べると，GSフォーマットに準拠したゲームソフトのBGMは10年後のGSフォーマットに準拠した音源でも演奏できます。

これがどのようなメリットを生むのでしょうか？ まず，ソフトメーカー側のメリットですが，GSフォーマットに対応することにより，多くのGSフォーマット準拠の音源に対応できるのです。それぞれの音源の知識はほとんど必要ないので，生産性は向上するはずです。これはものすごく画期的なことです。

また，ユーザー側から見てもメリットはあります。現状では，MIDI対応大半のゲームソフトやパソコンソフト用データ曲集はMT-32またはCM-64/32Lにしか対応していません。それだけの理由でCMシリーズを買う人が多いのです。しかし，ゲームソフトがGSフォーマットに準拠していれば，GSフォーマットに準拠した音源を選べるのです。

と，いう理由から今後多くのデータ曲集やゲームのGSフォーマットに対応すると思われます。Rolandとしてもソフトメーカーなどに対して積極的に働きかけているようです。

最後に，これからGSフォーマットが浸透するか否かは，やはりこれからのソフトウェア資産次第ということです。

規格というものは1機種で終わってはいけないものですから，次のGSフォーマット準拠の音源がどういう音源なのかとても楽しみです。

### どこまで使える？

残念ながら，現時点でもＧＳフォーマットは最終決定されていない部分があるらしい。建て前上は今回図表で載せたパラメータをすべてユーザーがいじっても構わないはずだが，エクスクルーシブやNRPNなどは今後も本当にサポートされるか疑問がある。

SC-55を使った投稿では，

レベル1：エクスクルーシブ，NRPNを除く
レベル2：ＧＳのフルスペック
レベル3：SC-55のフルスペック

のどの機能を使っているかを明示してほしい。

本来，ユーザーがなにも考えなくても互換がとれるようにするための規格だから，レベル2までは大丈夫でなくては困るのだが。　(S.N.)

**表3　GS音源のエクスクルーシブ**

**[SYSTEM PARAMETERS]**

| Address(H) | | | Parameter |
|---|---|---|---|
| 40 | 00 | 00 | WASTER TUNE |
| 40 | 00 | 04 | WASTER VOLUME |
| 40 | 00 | 05 | WASTER KEY-SHIFT |
| 40 | 00 | 06 | WASTER PAN |
| 40 | 00 | 7F | システムをリセット |

**[PATCH PARAMETERS]**

| Address(H) | | | Parameter |
|---|---|---|---|
| 40 | 01 | 00 | PATCH NAME 16 ASCII Characters |
| 40 | 01 | 10 | PARTIAL RESERVE-16 |
| 40 | 01 | 30 | REVERB MACRO |
| 40 | 01 | 31 | REVERB CHARACTER |
| 40 | 01 | 32 | REVERB PRE-LPF |
| 40 | 01 | 33 | REVERB LEVEL |
| 40 | 01 | 34 | REVERB TIME |
| 40 | 01 | 35 | REVERB DELAY FEEDBACK |
| 40 | 01 | 36 | REVERB SEND LEVEL TO CHORUS |
| 40 | 01 | 38 | CHORUS MACRO |
| 40 | 01 | 39 | CHORUS PRE-LPF |
| 40 | 01 | 3A | CHORUS LEVEL |
| 40 | 01 | 3B | CHORUS FEEDBACK |
| 40 | 01 | 3C | CHORUS DELAY |
| 40 | 01 | 3D | CHORUS RATE |
| 40 | 01 | 3E | CHORUS DEPTH |
| 40 | 01 | 3F | CHORUS SEND LEVEL TO REVERB |
| 40 | 1n | 00 | TONE NUMBER　CC:00 VALUE |
| 40 | 1n | 01 | P. C. VALUE |
| 40 | 1n | 02 | Rx. CHANNEL |
| 40 | 1n | 03 | Rx. PITCH BEND |
| 40 | 1n | 04 | Rx. CH PRESSURE(CAf) |
| 40 | 1n | 05 | Rx. PROGRAM CHANGE |
| 40 | 1n | 06 | Rx. CONTROL CHANGE |
| 40 | 1n | 07 | Rx. POLY PRESSURE(PAf) |
| 40 | 1n | 08 | Rx. NOTE MESSAGE |
| 40 | 1n | 09 | Rx. RPN |
| 40 | 1n | 0A | Rx. NRPN |
| 40 | 1n | 0B | Rx. MODURATION |
| 40 | 1n | 0C | Rx. VOLUME |
| 40 | 1n | 0D | Rx. PANPOT |
| 40 | 1n | 0E | Rx. EXPRESSION |
| 40 | 1n | 0F | Rx. HOLD 1 |
| 40 | 1n | 10 | Rx. PORTAMENTO |
| 40 | 1n | 11 | Rx. SOSTENUTO |
| 40 | 1n | 12 | Rx. SOFT |
| 40 | 1n | 13 | MONO/POLY MODE |
| 40 | 1n | 14 | ASSIGN MODE |
| 40 | 1n | 15 | USE FOR RHYTHM PART |
| 40 | 1n | 16 | PITCH KEY SHIFT |
| 40 | 1n | 17 | PITCH OFFSET FINE |
| 40 | 1n | 19 | PART LEVEL |
| 40 | 1n | 1A | VELOCITY SENSE DEPTH |
| 40 | 1n | 1B | VELOCITY SENSE DEPTH |
| 40 | 1n | 1C | PART PANPOT |
| 40 | 1n | 1D | KEY RANGE LOW |
| 40 | 1n | 1E | CC1 CONTROLLER NUMBER |
| 40 | 1n | 20 | CC2 CONTROLLER NUMBER |
| 40 | 1n | 21 | CHORUS SEND DEPTH |
| 40 | 1n | 22 | REVERB SEND DEPTH |
| 40 | 1n | 30 | TONE MODIFY 1<br>Vibrato rate |
| 40 | 1n | 31 | TONE MODIFY 2<br>Vibrato depth |
| 40 | 1n | 32 | TONE MODIFY 3<br>TVF cutoff freq |
| 40 | 1n | 33 | TONE MODIFY 4<br>TVF resonance |
| 40 | 1n | 34 | TONE MODIFY 5<br>TVF&TVA Env, attack |
| 40 | 1n | 35 | TONE MODIFY 6<br>TVF&TVA Env, decay |
| 40 | 1n | 36 | TONE MODIFY 7<br>TVF&TVA Env, release |
| 40 | 1n | 37 | TONE MODIFY 8<br>Vibrato delay |
| 40 | 1n | 40 | SCALE TUNING C |
| 40 | 1n | 41# | SCALE TUNING C # |
| 40 | 1n | 42# | SCALE TUNING D |
| 40 | 1n | 43# | SCALE TUNING D # |
| 40 | 1n | 44# | SCALE TUNING E |
| 40 | 1n | 45# | SCALE TUNING F |
| 40 | 1n | 46# | SCALE TUNING F # |
| 40 | 1n | 47# | SCALE TUNING G |
| 40 | 1n | 48# | SCALE TUNING G # |
| 40 | 1n | 49# | SCALE TUNING A |
| 40 | 1n | 4A# | SCALE TUNING A # |
| 40 | 1n | 4B# | SCALE TUNING B |
| 40 | 2n | 00 | MOD PITCH CONTROL |
| 40 | 2n | 01 | MOD TVF CUTOFF CONTROL |
| 40 | 2n | 02 | MOD AMPLITUDE CONTROL |
| 40 | 2n | 03 | MOD LF01 RATE CONTROL |
| 40 | 2n | 04 | MOD LF01 PITCH DEPTH |
| 40 | 2n | 05 | MOD LF01 TVF DEPTH |
| 40 | 2n | 06 | MOD LF01 TVA DEPTH |
| 40 | 2n | 07 | MOD LF02 RATE CONTROL |
| 40 | 2n | 08 | MOD LF02 PITCH DEPTH |
| 40 | 2n | 09 | MOD LF02 TVF DEPTH |
| 40 | 2n | 0A | MOD LF02 TVA DEPTH |
| 40 | 2n | 10 | BEND PITCH CONTROL |
| 40 | 2n | 11 | BEND TVF CUTOFF CONTROL |
| 40 | 2n | 12 | BEND AMPLITUDE CONTROL |
| 40 | 2n | 13 | BEND LF01 RATE CONTROL |
| 40 | 2n | 14 | BEND LF01 PITCH DEPTH |
| 40 | 2n | 15 | BEND LF01 TVF DEPTH |
| 40 | 2n | 16 | BEND LF01 TVA DEPTH |
| 40 | 2n | 17 | BEND LF02 RATE CONTROL |
| 40 | 2n | 18 | BEND LF02 PITCH DEPTH |
| 40 | 2n | 19 | BEND LF02 TVF DEPTH |
| 40 | 2n | 1A | BEND LF02 TVA DEPTH |
| 40 | 2n | 20 | CAf PITCH CONTROL |
| 40 | 2n | 21 | CAf TVF COTOFF CONTROL |
| 40 | 2n | 22 | CAf AMPLITUDE CONTROL |
| 40 | 2n | 23 | CAf LF01 RATE CONTROL |
| 40 | 2n | 24 | CAf LF01 PITCH DEPTH |
| 40 | 2n | 25 | CAf LF01 TVF DEPTH |
| 40 | 2n | 26 | CAf LF01 TVA DEPTH |
| 40 | 2n | 27 | CAf LF02 RATE CONTROL |
| 40 | 2n | 28 | CAf LF02 PITCH DEPTH |
| 40 | 2n | 29 | CAf LF02 TVF DEPTH |
| 40 | 2n | 2A | CAf LF02 TVA DEPTH |
| 40 | 2n | 30 | PAf PITCH CONTROL |
| 40 | 2n | 31 | PAf TVF CUTOFF CONTROL |
| 40 | 2n | 32 | PAf AMPLITUDE CONTROL |
| 40 | 2n | 33 | PAf LF01 RATE CONTROL |
| 40 | 2n | 34 | PAf LF01 PITCH DEPTH |
| 40 | 2n | 35 | PAf LF01 TVF DEPTH |
| 40 | 2n | 36 | PAf LF01 TVA DEPTH |
| 40 | 2n | 37 | PAf LF02 RATE CONTROL |
| 40 | 2n | 38 | PAf LF02 PITCH DEPTH |
| 40 | 2n | 39 | PAf LF02 TVF DEPTH |
| 40 | 2n | 3A | PAf LF02 TVA DEPTH |
| 40 | 2n | 40 | CC1 PITCH CONTROL |
| 40 | 2n | 41 | CC1 TVF CUTOFF CONTROL |
| 40 | 2n | 42 | CC1 AMPLITUDE CONTROL |
| 40 | 2n | 43 | CC1 LF01 RATE CONTROL |
| 40 | 2n | 44 | CC1 LF01 PITCH DEPTH |
| 40 | 2n | 45 | CC1 LF01 TVF DEPTH |
| 40 | 2n | 46 | CC1 TVA CUTOFF CONTROL |
| 40 | 2n | 47 | CC1 LF02 RATE CONTROL |
| 40 | 2n | 48 | CC1 LF02 PITCH DEPTH |
| 40 | 2n | 49 | CC1 LF02 TVF DEPTH |
| 40 | 2n | 4A | CC1 LF02 TVA DEPTH |
| 40 | 2n | 50 | CC2 PITCH CONTROL |
| 40 | 2n | 51 | CC2 TVF CUTOFF CONTROL |
| 40 | 2n | 52 | CC2 AMPLITUDE CONTROL |
| 40 | 2n | 53 | CC2 LF01 RATE CONTROL |
| 40 | 2n | 54 | CC2 LF01 PITCH DEPTH |
| 40 | 2n | 55 | CC2 LF01 TVF DEPTH |
| 40 | 2n | 56 | CC2 LF01 TVA DEPTH |
| 40 | 2n | 57 | CC2 LF02 RATE CONTROL |
| 40 | 2n | 58 | CC2 LF02 PITCH DEPTH |
| 40 | 2n | 59 | CC2 LF02 TVF DEPTH |
| 40 | 2n | 50 | CC2 LF02 TVA DEPTH |

ようこそここへC言語

# ポインタって何だろう（後編）

［第12回］

Nakamori Akira
中森 章

先月に引き続きポインタのお話です。先月で一応の基礎知識は身についたことと思いますので，今月はやや応用編に近い解説になっています。この2カ月で，難しいと思っていた人もポインタを理解できたのではないでしょうか。

映画「おもひでぽろぽろ」を観て，2/3個のりんごを1/4で割ったときの意味を考え込んでしまった中森章です。たしかに分母と分子をひっくりかえして掛け算してやれば分数の割り算はできるのですが，直感に訴えてこない概念はもやもやした感じを残していつまでもあとを引きますね。C言語のポインタの概念も分数の割り算のような気分で受け止めている人は結構いるのかもしれませんね。

さて，今回のテーマはポインタの続きです。関数へのポインタを中心に話を進めますが，その前にポインタへのポインタの具体例としてmain関数への引数について説明したいと思っています。

## 文字列を要素とする配列

ポインタ変数に格納されるアドレス値は符号なしの整数値とみなすこともできます。このためポインタ（アドレス値）を要素とする配列を考えることもできます。このような配列があることは前回でも少し話しておきましたが，この具体例として文字列を要素とする配列を考えましょう。たとえば，文字列を要素とする1次元配列，

```
char *fruits [ ]= {
        "Kiwi", "Papaya",
        "Mango", "Durian"
};
```

を考えます。ここで，文字列そのものはアドレス値になりますから，fruitsという配列は4つのアドレスを要素として持つことになります。すなわち，

```
fruits [0]  は “Kiwi”    というアドレス
fruits [1]  は “Papaya”  というアドレス
fruits [2]  は “Mango”   というアドレス
fruits [3]  は “Durian”  というアドレス
```

となります。いうまでもなく，宣言のchar ＊という表現は，fruitsの各要素がchar型データを指し示すポインタ（アドレス値）であることを意味しているのです。このような配列では各要素（ポインタ）の指し示す先が実際の文字の並びになります。

さて，このような配列fruitsの特定の要素を指し示すポインタ変数はどう宣言するのでしょうか。これは，ポインタ変数の内容（指し示すもの）がさらにポインタである場合ですから，

```
char **p;
```

でよかったのでしたね（＊がひとつで1回のポインタによる参照です）。このときポインタPが配列fruitsの先頭アドレスを保持していれば，

```
p      →   &fruits [0]
p+1    →   &fruits [1]
p+2    →   &fruits [2]
p+3    →   &fruits [3]
```

あるいは，

```
*p      →   fruits [0]
*(p+1)  →   fruits [1]
*(p+2)  →   fruits [2]
*(p+3)  →   fruits [3]
```

という対応を取ることができます(復習ですよ)。このときのポインタと要素の関係を図1に示しておきます。ポインタへのポインタという概念を頭に叩き込んでくださいね。

ところで，ポインタ変数Pによってfruits［2］（すなわち“Mango”)の4番目の文字(fruits[2][3])を参照したい場合は，

```
*(*(p+2)+3)
```

という式になります。これは＊演算子を用いた式ですが，これを［ ］演算子を使って書き換えると，

```
*(p+2) [3]
```

または，

```
p [2] [3]
```

となります。これはポインタ変数Pが2次元配列（の名前）と同等であることを示しています。どちらも，参照を2回行う（＊＊あるいは［ ］［ ］という操作）ことによって最終的なデータ（いまはchar型データ）に行き当たるようになっています。

しかし，ポインタへのポインタと2次元配列の決定的な違いは，1回目の参照（＊あるいは［ ］という操作）で行き当たるデータの性質です。それが1次元配列かポインタかという違いがあるのです。2次元配列の場合は

1回目の参照で行き当たるデータは，ある大きさ（配列宣言時の［ ］［ ］において，2つめの［ ］内に書かれる要素数によって決まる）を持った1次元配列です。この1次元配列がさらに1次元配列の要素となってメモリ上に連続して並んでいます（これが2次元配列というものでしたね）。しかし，ポインタへのポインタの場合は1回目の参照ではポインタに行き当たります。このポインタが1次元配列の要素となってメモリ上に連続して並んでいます。そして，各要素であるポインタの指し示す先が最終的なデータです。あきらかに，2次元配列とポインタへのポインタでは，1回目の参照で行き当たるデータがメモリ上で占有するバイト数が異なっていますね。

配列fruitsを，

```
    char fruits [ ] [7] = {
            "Kiwi", "Papaya",
            "Mango", "Durian"
    };
```

と2次元配列で記述した場合のメモリの様子を図2に示しておきますので，図1と比較してみてください。どちらの場合も意味的には文字列を要素とする配列ですが，メモリの使用効率はかなり違っていますね。2次元配列で宣言した場合は，ひとつの文字列に対してある程度の大きさ(最大の長さの文字列が格納できるだけの大きさ)のメモリ領域を必ず確保しなければなりません。たとえば，上の例（2次元配列のfruits)ではひとつの文字列用に7文字分の大きさ（7バイト）を確保しています。これは，文字列の終わりを示すヌル文字（char型データの0）を含めて“Papaya”や“Durian”という文字列を格納するための最小の大きさです。このとき，“Kiwi”や“Mango”といった6文字よりも短い長さの文字列を格納する場合は領域が無駄になってしまいます。一方，ポインタへのポインタの場合は，各文字列を指し示すポインタのために必ず4バイトのメモリ領域が確保されることになりますが，それが指し示す文字列の長さは同一である必要はなくメモリに無駄は生じません。このため，ポインタのために4バイト必要になるとはいえ，文字列の配列を扱う場合はポインタへのポインタで表現したほうが一般的にはメモリの使用効率がよくなります（2次元配列より効率が悪くなるのはどういう場合かわかりますね)。

ところで，文字列はポインタ（アドレス値）であると理解するのが自然ですから，上の2次元配列の宣言が，

```
    char fruits [ ] [7] = {
            {'K','i','w','i', 0, 0, 0},
            {'P','a','p','a','y','a',0},
            {'M','a','n','g','o', 0, 0},
            {'D','u','r','i','a','n',0}
    };
```

と解釈されてコンパイルされるのは不思議な気がしませんか。このとき文字列は，ポインタではなく，文字を要素とする1次元配列そのものとしてコンパイルされているのです。これは，根本的には，

```
    char *str1  = "Thank you!";
```

と，

```
    char  str2 [ ] = "Thank you!";
```

との宣言の解釈の違いに帰着する問題です。文字列を初期値として与えるときは，その格納先によって文字列の意味が変わってくるのです。このように，C言語では文字列は時と場合によってポインタになったりchar型の1次元配列になったりするので戸惑うこともありますが，そのうち慣れてきますからまあそんなものだと思っておいてください[1]。

**図1　文字列へのポインタを要素とする配列**

---

1）文字列に関してはいろいろ不可解である。

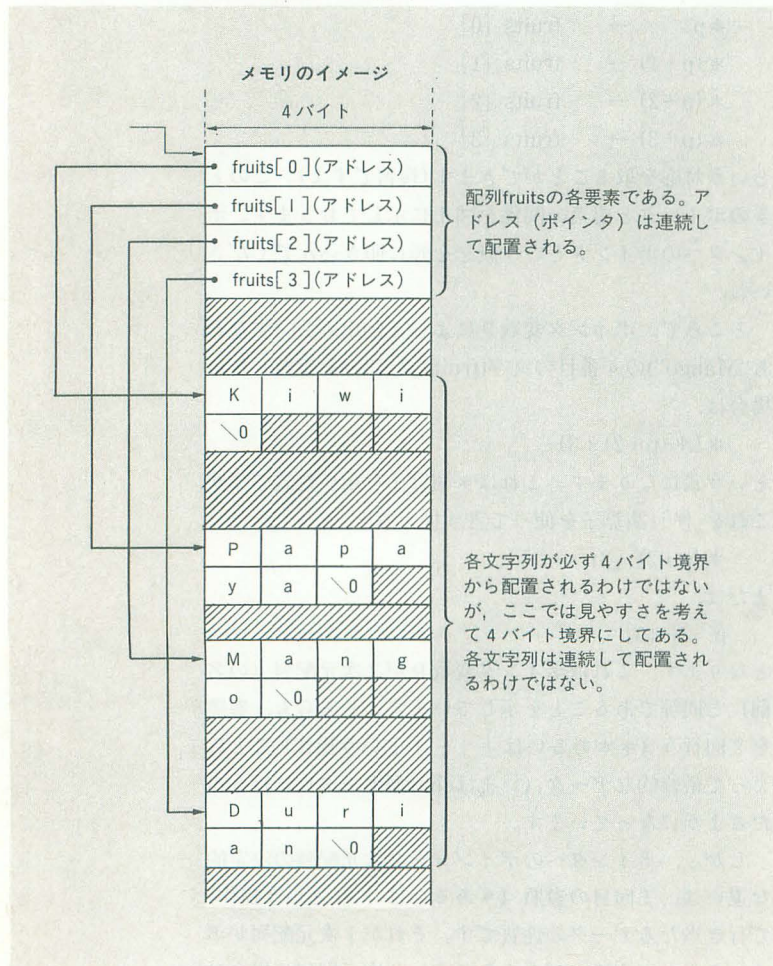
```
char str [ ] ="abc";
```

という宣言は要素が'a'，'b'，'c'，0であるchar型の配列を宣言する。配列の添字を与えた宣言では，

```
char str [4] ="abc";
```

となる。ところがC言語の文法では，

```
char str [3] ="abc";
```

も許される。通常，配列の初期化で宣言よりも多い要素数を指定するとエラーになるはずだが，文字列の最後のヌル文字（0）の扱いは例外である。このとき文字列の最後のヌル文字は配列の要素に含まれない。

## main関数への引数

ポインタへのポインタの話が出たところで，それと非常に関係のある（というより，そのもの）main関数への引数について説明しましょう。main関数とはプログラムの中で最初に実行されるあの関数のことです。通常は，

```
main( )
{
    .........
}
```

というようにプログラムが書かれるので，main関数に引数がないものと思いがちです。しかし，これはせっかく与えられた引数をプログラムの側で無視しているだけのことなのです。

一般には，関数を呼び出すときに引数を与えるのはその関数を呼ぶプログラムです。ところが，main関数はC言語では一番最初に実行されますから，どこかの関数から呼び出されるというものではありません[2]。大雑把には，main関数に引数を与えるのはOS（Human68k）と思っておけばよいでしょう。この意味で，main関数への引数は通常の関数の引数とは少し意味合いが違っています。それはプログラムとOSとのインタフェイスを行うために用意されているものなのです。

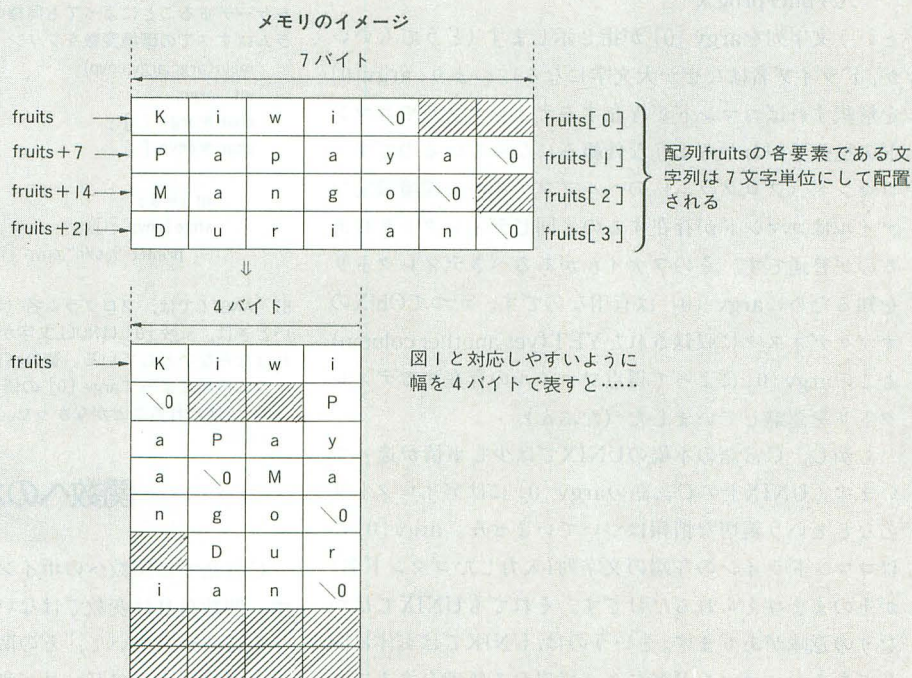
C言語はシステムを記述するための言語として誕生しました。C言語はOSの核を記述するためにも使用されますが，その用途のほとんどはOSに付属するコマンドを記述するためのものです。このコマンドの作成を容易にするのがmain関数への引数なのです。コマンドは通常，

```
dir /N /R
```

のように，コマンド名（dir）と引数（/Nや/R）を対にしてコマンドラインから入力します。あるプログラムがこのようなコマンドとして機能するためには，コマンドラインに与えられた引数の情報を知る必要があります。そして，C言語ではこのような情報を知るための手段がmain関数への引数として実現されているのです。K&RやANSI Cではmain関数の引数として2つの引数を規定しています。main関数への第1引数はargc（argument count：引数の数の意）と呼ばれ，プログラムを呼び出したコマンドラインの引数の個数（コマンド名を含む）を示します。どのようなコマンドを入力するときもコマンド名は必ず入力しますから，argcの値は必ず1以上になります。上のdirの例ではargcの値は3になります。そして，main関数の第2引数はargv（argument vector：引数の配列の意）と呼ばれ，各引数（コマンド名を含む）を表す文字列へのポインタの配列となっています[3]。argvのイメージとしてはコマンドラインに与えられた各文字列を要素とする配列です。上の例でいえば，

```
argv [0]  →  "dir"
argv [1]  →  "/N"
argv [2]  →  "/R"
argv [3]  →  NULL   （0のことです）
```

という対応になります。このとき，argvはchar型を要素とする2次元配列ではなく，char型へのポインタを要素

**図2　文字列を要素とする配列**

とする配列となっているのです。

以上のことから，コマンドラインの情報をプログラムで受け取るためにはmain関数を，

```
main(argc,argv)
int  argc;
char *argv [ ];
{
    ………
}
```

などと宣言しておけばよいことがわかると思います[4]。ためしに，すべての引数の内容（コマンド名を含む）を知りたければ，上の宣言をしたあと，

```
for(i=0；i<argc;i++)printf("%s¥n",argv[i])；
```

あるいは，

```
i=0；
while(argv [i]) printf("%s¥n",argv [i++])；
```

というプログラムを実行すればよいでしょう（変数iは宣言していなければなりませんよ）。

ところで，あるコマンドを作成するときにコマンドラインから入力されるコマンドへの引数が必要になることは容易に想像ができます。しかし，コマンド名(argv[0])がなぜ必要なのかわかりますか。これは単なるオマケなのでしょうか[5]。もちろんそうではありません。argv[0]はそれなりに重要な使用法があるのです。

上のプログラムを実行した人ならわかると思いますが，Human68kではargv [0] で示されるコマンド名にはそのプログラムが存在するディレクトリの名前が付加されています。たとえば，a:¥binというディレクトリにあるprog.xというプログラムを実行すると，

A:¥bin¥prog.x

という文字列をargv [0] が指し示します（どうでもいいが，ドライブ名はなぜか大文字になる)。つまり，argv[0]を解釈すればコマンドが存在するディレクトリやコマンドの拡張子がわかるような仕組みになっているのです。コマンド（プログラム）のヘルプファイルや環境設定ファイルはコマンドが存在するのと同じディレクトリにあるのが普通です。そのファイルがあるべきディレクトリを知るためにargv [0] は有用なのです。かつてOh!Xのオマケディスクに収録されたYET(yet another column)もこのargv [0] によって得点ファイルの存在するディレクトリを認識していました（たぶん)。

しかし，C言語の本場のUNIXでは少し事情が違っています。UNIX上のC言語のargv [0] にはディレクトリ名などという親切な情報はついていません。argv [0] にはコマンドラインの左端の文字列(入力したコマンド名)がそのまま与えられるだけです。それでもUNIXではかなりの意味があります。というのは，UNIXでは実体は同じであるのにコマンド名によって異なる処理をするプログラムが存在するからです。詳しい説明は省略しますが，UNIXではln(リンク)というコマンドによってあるプログラムを別の名前で実行できるように指定することができます。プログラムの側ではそのプログラムが呼び出された名前（argv [0]）にしたがって処理を切り替えるのです。たとえば，UNIXにはファイルを圧縮するためのcompressというコマンドがあります。圧縮したファイルを解凍するためのコマンドはuncompress, 圧縮したファイルを画面に表示するコマンドはzcatです。実はこれらのファイルの実体はすべて同じものなのです（ソースプログラムはcompress.cというひとつがあるだけ)。圧縮・解凍に使用するキーデータは共通ですから，そのデータのテーブルをプログラムごとに持つのは非効率的です。そこで，３つの異なる機能がひとつのプログラムにまとめられているのです。これと同様にUNIXのエディタのexとviも実体がひとつであることが多いようです。

---

2) main関数は正確にはスタートアップルーチンから呼び出される。X68000用のC言語ではスタートアップルーチンはライブラリの中にあり，リンカによってユーザーのプログラムと結合される。なお，スタートアップルーチンは通常アセンブリ言語で記述されている。実は，main関数への引数をセットするのもスタートアップルーチンの役目である。

3) argcやargvは慣例的な名前。これらは仮引数の名前なので名前はなんでもよい。ただし，これ以外の名前を使うプログラムはまず見かけない。と思っていたのだけど，先日cshのソースを読んでいたらargc, argvとは違う名前を使っていたなあ。

4) K&RやANSI Cでは特に記述されていないが，多くの場合C言語のmain関数は３つの引数を持つ。最初の２つはargcとargvであり，第３の引数が環境変数へのポインタである。これは通常envp（environ ment pointerの略）と呼ばれている。envpの形式はargvと同じく，文字列（char型へのポインタ）を要素とする配列である（最終要素はNULL)。そして，その各要素は，

"環境変数名=……"

という文字列となっている。C言語では環境変数の値を取り出すためのgetenvというライブラリ関数が用意されているが，envpの内容をサーチすることによっても同様の機能を実現できる。次のプログラムはすべての環境変数をプリントする。

```
main(argc,argv,envp)
int  argc;
char *argv [ ];
char *envp [ ];
{
    int i=0;
    while(envp [i])
        printf("%s¥n",envp [i++]);
}
```

5) ANSI Cでは，プログラム名（コマンド名）が環境から得られないときは，argv [0] はNULL文字からなる文字列（""のこと）でなければならないとしている。昔のパソコンのCコンパイラではコンパイラの種類によってargv [0] の値はばらばらだった。つまり，argv [0] は使われることがなかった。

## 関数へのポインタ

C言語では関数へのポインタを定義することができます。関数自身は変数ではないのですが，その実体はメモリ上に置かれていて，その関数の入り口には関数名と１対１に対応するアドレスが割り付けられています。これはイメージ的には配列と同じです。配列の先頭アドレス

を配列名として認識するように，コンパイラは関数の先頭アドレスを関数名として認識するのです。つまり，関数名はアドレスを示す定数値として扱われているのです。

次のプログラムを見てください。

```
short main [ ]= {
    0x487a, 0x0006, 0xff09, 0xff00,
    0x6865, 0x6c6c, 0x6f2c, 0x2077,
    0x6f72, 0x6c64, 0x0d0a, 0x0000
};
```

ちょっと見にはmainというshort型データの配列を宣言しているだけです。しかし，このプログラムはこのままで実行できてしまうのです（何が起きるかコンパイルして実行してみてくださいね）。

これは，mainという配列が，C言語のプログラムで一番最初に実行されるmain関数と間違われて実行されてしまうためなのです。こういう例を見ると，配列も関数もたいして違いがないことがイメージできるのではないでしょうか。どちらもメモリ上に格納されたデータの並びです（これはコンパイル後の話）が，関数の場合はそれが実行可能な機械語の命令であるという点が異なるのです。

関数名はアドレス値ですから，それはポインタ変数に代入したり，別の関数への引数としたり，配列の要素とすることができます。そして関数を示すアドレス値（関数名）を格納しておき，あとで間接的に関数を呼び出すための変数が関数へのポインタなのです。それでは例によって，その宣言と使用方法について見ていくことにしましょう。

## ●関数へのポインタの宣言

関数を指し示すポインタの宣言は少し変則的です。ポインタですから変数のときと同じく＊を用いて宣言することになりますが余分に（と）が必要です。たとえば，int型データを戻り値とする関数ｆの宣言は，

```
int f( );
```

でした(覚えてますよね)。これは関数の本体を定義しているのではなく，int型を戻り値とする関数ｆが存在するということをコンパイラに教えるための宣言です。同様にint型データへのポインタを戻り値とする関数fpの宣言は，

```
int *fp( );
```

です。それに対しint型を戻り値とする関数を指し示すポインタ（簡単に関数へのポインタと呼ぶ）fpの宣言は，

```
int (*fp)( );
```

となります。これは，上の２つの宣言とは異なり，関数が存在することをコンパイラに教えるものではありません。fpというポインタによって間接的に呼び出される(fpが保持するアドレスを呼び出す)関数の戻り値がint型であることを示しています。呼び出す先の関数が実際に存在するかどうかはわかりません。

ところで，関数へのポインタの宣言は＊と関数名が（と）で囲まれているのがキーポイントです。演算子の優先順位を考えると，関数呼び出しを示す（ ）はポインタ参照の＊よりも優先順位が高くなっています[6)]。したがって，

```
*fp( )
```

は，

```
*(fp( ))
```

と読み下すことで，関数の戻り値がポインタであることがわかります。一方，

```
(*fp)( )
```

はfpが指し示すもの(これが＊fp)によって呼ばれる関数という意味なのです。

さて，ANSI Cでは関数の宣言にプロトタイプというものが採用されました。これは，特に関数の引数の数とデータ型をチェックするための機構です（5，6月号のこの連載の「関数って何だろう」の回を参照してね）。もちろん，関数へのポインタの宣言においてもプロトタイプを宣言することができます。この場合はポインタの宣言の右端にある（ ）の中に引数のプロトタイプを記述することになります。たとえば，

```
double (*fp)(int,float);
```

という記述は，ポインタfpによって間接的に呼ばれる関数は２つの引数を持ち，第１引数がint型，第２引数がfloat型であることを宣言します。いうまでもなく，関数の戻り値はdouble型です。

---

6) 演算子の優先順位とは式が実際に計算されるときの実行順序であって配列や関数などの宣言時の記号とは直接は関係ない（はずである）。ただし，配列や関数などの宣言で使われる（演算子と同じ）記号も意味的には演算子と同じ性質を持つので，結合の強さも同じと考えてよいだろう。

## ●関数へのポインタの初期化

何度もいいますが，どんなポインタも初期化しなければ意味がありません。関数へのポインタは基本的には関数名で初期化します（ほかのポインタの値を代入することもできますが）。

たとえば，ｆという名前の関数があってfpが関数へのポインタ変数である場合，

```
fp = f;
```

によって，fpに値を与えることができます。これは変数へのポインタを配列名で初期化するのと似ています。ただし，このときコンパイラがｆを関数と認識していなければ何が起こるかわかりません。ｆは通常の変数の名前かもしれませんし，配列名かもしれません。そんなとき変数や配列に格納されている（命令としては）意味のないデータが関数として呼び出されて実行されてしまいます(普通は暴走ですね)。関数へのポインタとして無意味なものを代入しないようにするのはプログラマの責任です(一応，代入するポインタの型が一致しないという警告メ

ッセージは出るのですがエラーにはならない)。

なお，関数へのポインタに関数名を代入するためには，あらかじめfという名前が関数名であることをコンパイラに教えてやることが必要です。そのためには，初期化の代入よりも先に関数fの本体の定義を記述するか，fが関数だよという宣言をする必要があります。

```
int f( );
```

などと関数fの戻り値のデータ型を宣言する（戻り値の型だけで本体の定義はない）ことが，fが関数だよという宣言になります。

さて，関数へのポインタを宣言と同時に初期化するためにはどうすればよいのでしょう。これは，通常のポインタの初期化と同様に，宣言のあとに=を続けて行えばよいのです。具体的には，

```
int (*fp)( )=f;
```

などとなります。少し奇異な感じを受けます（=の前に( )があっていいのだろうか）が，本当にこれでよいみたいです。かえって，考えすぎて，

```
int (*fp=f)( );
```

などとすると文法エラーになってしまいますから気をつけましょう。

ところで，これまで関数は配列と同じようなものだという説明をしてきました。関数へのポインタに初期値を代入する場合は関数の実体が存在するアドレスが必要になります(これが関数名ですね)。このとき，関数のアドレスを取り出す方法も配列の場合と非常によく似ているので，ここに付け加えておきましょう。つまり，関数は関数名そのものがアドレスになりますが，関数名に&を付けたものも同じアドレスになります。関数fがあるとき，

```
f
```

と，

```
&f
```

はどちらも関数のアドレスを示しているのです。配列の場合は先頭の要素に&を付けたものも先頭アドレスになることができました。さすがに，関数には先頭の要素という概念がないのでそのような方法でアドレスを取り出すことはできません。

### ●関数へのポインタを使った関数の呼び出し

関数へのポインタを用いて関数を呼び出す方法は，K&Rの時代とANSI Cの時代では少し異なっています。もちろん，ANSIのほうが上位コンパチで拡張されているのですが，まずはK&Rでの呼び出し方法を説明することにしましょう。

いまfpが関数へのポインタとして宣言され，何か適当な関数のアドレスで初期化されているとします。このとき，そのポインタが指し示す関数を呼び出すためには，

```
(*fp)( )
```

という記述をします。これによってfpの指し示す関数が間接的に呼ばれるのです。これは関数呼び出しに引数を与えない場合です。もし，関数呼び出し時に引数を与える必要があれば，

```
(*fp)(a,b,c)
```

などと( )内に引数を記述すればいいでしょう(a，b，cが引数ですよ)。関数のポインタによって関数を呼び出すといっても，それは通常の関数の呼び出しと同じです。

```
(*fp)(1,2);
```

のように単体で呼び出したり，

```
x=(*fp)(1,2)+3;
```

のように式の一部として使用することができます。

さて，ANSI Cではポインタによる関数の呼び出しのために別の表現が導入されました。それはポインタによって関数を呼ぶときに*とポインタを囲む( )を省略するという記述です。

つまり，従来，

```
(*fp)( );
```

と記述していた表現を，単に，

```
fp( );
```

と表現できるようになったのです。これは普通の関数呼び出しとまったく同じ表現ですし，従来の方式と比べると非常に見やすくなっています。おそらく，配列名と関数名の類似性を考慮して，配列を指し示すポインタに[ ]演算子をつけることで要素をそのまま取り出すことができるように，関数へのポインタも( )を付けることでそのまま関数を呼び出せるようにしたのでしょう。fpはすでに宣言されていて，コンパイラは関数へのポインタであることを知っていますから，これでも混乱は起きないのです。

関数の呼び出しに*と( )が不要になったのと同時にANSI Cでは関数の仮引数にある関数へのポインタの表現も簡略な表現が許されるようになりました。従来，関数へのポインタを引数とする関数の宣言は，

```
func(fp)
int (*fp)( );
{
    ………
};
```

となっていましたが，ANSI Cでは，

```
func(fp)
int fp( );
{
    ………
};
```

という表現もできるようになったのです[7)]。ただし，ANSI Cでも関数の引数でない場合の関数へのポインタの宣言は，簡略した表現をすることはできません。簡略した表現では関数へのポインタであるのか単に関数の戻り値を宣言しているのかの区別がつかなくなるので，当

然といえば当然ですね。

---

7) とはいえ，この表現が許されるコンパイラは少ない。GCCではコンパイルできるがXCではコンパイルできない。あるコンパイラがANSI Cに準拠しているかどうかを知るためのテスト項目のひとつにできそうだ。

### ●関数へのポインタへのキャスト

関数へのポインタの宣言は普通の変数や配列の宣言とは少し毛色が変わっていました。ポインタを関数へのポインタにキャスト（型変換）するときも毛色が違っていますから覚えておきましょう。関数へのポインタへのキャストは，

```
(戻り値 (*)( ))
```

という記号を変数や定数の前につけることによって行います。たとえば，変数xをdouble型を戻り値とする関数へのポインタにキャストするためには，

```
(double (*)( ))x
```

という表現になります。これは知っていないとなかなか思いつかない表現ではないでしょうか。

さて，関数へのポインタへのキャストはデータの型変換をコンパイラに伝えるために必要です。ただし，関数へのポインタに代入する値として明らかに不正なデータを与えるのでなければ，コンパイラのほうで勝手に入力データの型変換をしてくれる（警告メッセージは出る）ので，関数へのポインタへのキャストが（実質的に）必要になる場面はそれほど多くありません（ちゃんと型変換を明示したほうがよいことには変わりないが)。このようなキャストが本当に必要になるのは，int型変数が保持している値のアドレスにある関数を呼び出す場合，アドレス値を定数で指定して関数を呼び出す場合だと思われます[8)]。前者は，

```
((void (*)( ))x)(1,2);
```

などという式（xはint型変数)，後者は，

```
((void (*)( ))0x6800)(1,2);
```

などという式によって実現することができますが，このようなプログラムを書くことは一生のうちで1回か2回ぐらいしかないでしょう。他人の書いたプログラムを読むための基礎知識ぐらいと思っておいてかまいません。

ところで，上の例ではどちらの場合もキャストした関数へのポインタをさらに( )で囲んでいることに注意しましょう。これは関数呼び出しを行う( )という（引数を与える）演算子のほうが，キャストを示す( )という演算子よりも優先順位が高いためです。

---

8) 関数へのポインタにキャスト可能なデータは実質的にポインタと同等な整数(char型，int型など)，配列名，関数名，アドレス演算(&)の結果などである。float型やdouble型のデータをそのまま関数へのポインタにキャストすることはできない。

### ●関数へのポインタの効能

これまで関数へのポインタを長々と説明してきましたが，関数へのポインタがなぜ必要なのかを考えてみましょう。

昔，特に8ビットパソコン用のCコンパイラはアセンブリ言語（機械語）との結合が非常に厄介でした。そこで，機械語のプログラムとの結合が必要な場合は，配列の中に機械語コードを格納しておき，そこを呼び出すというテクニックがしばしば用いられていました。X68000では，

```
short code [ ]= {
    0x2f2f, 0x0004, 0xff09,
    0x588f, 0x4e75
};

main( )
{
    void (*func)( )=(void (*)( ))code;

    (*func)  ("hello, world¥r¥n");
    (&*func) ("don't break a peace¥r¥n");
    func     ("make our own destiny¥r¥n");
}
```

というようなプログラム例になるでしょうか（配列codeの中身は引数で与えられた文字列を画面にプリントする機械語プログラムです)。ただし，このような使い方をC言語の開発者（リッチーなど）が考えていたかどうかは疑問です。なぜなら，まっとうなCコンパイラなら最初からアセンブリ言語とのリンクを行うためのasm文とか#asm疑似命令が用意されているのが普通だからです。配列の中にわざわざ機械語を書き込むような苦労は必要ありません。

関数へのポインタはプログラムで呼び出すべき関数の実体が不明な場合に，とりあえず関数呼び出しを行うための手段に使うことができます。おそらく，このような使い方が関数へのポインタの「正しい」使い方だと思われます。

たとえば，XCのライブラリにはqsortというクイックソートという関数が用意されています(qsortはこの連載でも何度か取り上げましたね)。これはいろいろな形式の入力データをクイックソートするための汎用的な関数です。ソートを行うためにはデータの大小関係が決まってなければなりません。データが数値ならば大小関係は明らかですが，データが文字列であったりすると文字コードに従ったソートを行うか辞書的順序でのソートを行うかで大小関係が違ってきます[9)]。つまり，大小関係が与えられないとソートはお手上げです。

そこで，qsort関数では大小関係を決定するための関数（へのポインタ）を引数で与えるようになっています。qsort関数の内部ではその引数（関数へのポインタ）が指し示す関数を呼び出して大小関係を判別するようにプログラムされているのです。

qsort関数に限らず，関数へのポインタのこのような使用方法は，関数の実体を次々と差し替えて利用することになる，方程式の解を求めるプログラム，数値積分を行うプログラム，関数の値を画面上にプロットするプログラムなど多くのプログラムで必要になってくるでしょう。

9) 辞書的順序とは，たとえば，"a"，"b"，"A"，"B"という文字列があったら"a"，"A"，"b"，"B"の順序になるような順序のこと。これは辞書に載っている単語の順序に一致する。文字コードの順序なら"A"，"B"，"a"，"b"の順になる。

### リスト1 関数へのポインタを利用するプログラム

```
 1: /*
 2:        m a i n関数への引数を使用するプログラム例
 3:
 4:        (標準入力から入力されるタブを空白に変換する)
 5: */
 6: char LINE [1024+1];
 7: char LINE2[1024+1];
 8: int  TABPOS=8;   /* 規定値：8カラムにそろえる */
 9:
10: expand()       /* タブを空白に変換する */
11: {
12:        int i,j,k;
13:        int ch,fill;
14:        for(i=0,j=0;i<1024;i++){
15:            ch=LINE[i];
16:            if(ch==0) break;
17:            if(ch=='¥t'){
18:                fill=TABPOS-(j % TABPOS);
19:                for(k=0;k<fill;k++)
20:                    LINE2[j++]=' ';
21:            }
22:            else{
23:                LINE2[j++]=ch;
24:            }
25:        }
26:        LINE2[j++]=0;
27:        strcpy(LINE,LINE2);
28: }
29:
30: main(argc,argv)
31: int argc;
32: char      **argv;
33: {
34:        int i;
35:
36:        for(i=0;i<argc;i++){          /* "-t","-T","/t","/T"を判別 */
37:            if(argv[i][0]!='-' && argv[i][0]!='/') continue;
38:            if(argv[i][1]!='t' && argv[i][1]!='T') continue;
39:            TABPOS=atoi(argv[i]+2); /* atoi で数値への変換 */
40:            if(TABPOS==0) TABPOS=8;  /* 0を指定されると困るので */
41:        }
42:        while(gets(LINE)!=0){             /* 1行入力 */
43:            expand();                     /* 変換      */
44:            printf("%s¥n",LINE);          /* 1行出力 */
45:        }
46: }
```

### リスト1の実行結果

```
(a)入力ファイル (インデントはすべてタブ)
/*
        サンプルプログラムだよ
*/
#include <stdio.h>
main()
{
        printf("Hello, world.¥n");
}

(b)出力結果 (コマンドラインで -t1 を指定)
/*
 サンプルプログラムだよ
*/
#include <stdio.h>
main()
{
 printf("Hello, world.¥n");
}
```

### ◆基礎力を高めよう

**設問1** プログラムで次の1)～4)の変数宣言がされている場合，それぞれの宣言の意味の違いを答えてください。

1) int p [ ] ;
2) int *q;
3) int r [ ] = {0} ;
4) int *s= {0} ;

**設問2** プログラムで次の1)～4)の変数宣言がされている場合，それぞれの宣言の意味の違いを答えてください。

1) int (*fpa [ ])( ) ;
2) int (*fpb [3])( ) ;
3) int (*fpc( )) [ ] ;
4) int (*fpd( )) [3] ;

(解答は120ページ)

## ポインタを利用するプログラム

それでは，今回説明した2つの事項に関してサンプルプログラムを示しましょう。第1はmain関数への引数を利用するプログラム，第2は関数へのポインタを利用するプログラムです。

### ●main関数への引数を利用するプログラム

コマンドラインから入力される文字列は「コマンド名」，「オプション」，「ファイル名」などです。gccでC言語のプログラムをコンパイルするための，

gcc -O dhry.c -DREG=register

というコマンドラインを考えれば，“gcc”がコマンド名，“-O”と“-DREG=register”がオプション，“dhry.c”がファイル名になります。もっとも，コマンドに対する引数部分がどのような意味（オプションかファイル名かそれ以外か）になるかは，コマンドが引数をどのように解釈するかによっていますから，一般的にはオプションとファイル名の切り分けはできません。多くのコマンド（プログラム）では引数の最初の文字が“-”あるいは“/”であるときに，その引数をオプションとして認識するように設計されているようです。それ以外の引数はファイル名であるとすることが多いようです。

リスト1は，テキストファイル内にあるタブを空白に置き換えるプログラムです。このプログラムはタブの停止位置（オプションの指定）をコマンドラインで指定し，標準入力から与えられるテキストファイルのタブを空白に置き換え，結果を標準出力に出力します。このとき，コマンドラインでは（コマンド名と）タブの停止位置を入力するだけですから，

expand 10 < infile

のように，直接数値を指定する方法も考えられます（コマンド名をexpand，入力ファイルをinfileと仮定しています）。ここでは，オプションらしく見せるために，-tあるいは/t（tは大文字でもよい）に続けて数値が指定され

たときだけ，その数値をタブの停止位置と認識するようにしています。つまり，コマンドラインでの入力は，

expand -t10 < infile

などのようになるでしょう。

さて，iが0から(argc－1)の間の整数であるとき，argv [i] はコマンドラインに入力された文字列へのポインタですから，

argv [i] [j]

がargv [i] が示す文字列のｊ番目（0から数えて）の文字となります。そこでこのプログラムの場合は，コマンドの引数がオプションかどうかを調べるためには，まず，

argv [i] [0]

が“-”か“/”であることを調べて，その次に，

argv [i] [1]

が“t”か“T”であるかを調べればよいのです（これ以外にも調べ方はいろいろあります)。これによって文字列の先頭が“-t”，“/t”，“-T”，“/T”のどれかであることがわかれば，それ以降に書かれている数値は，argv[i]の3文字目が格納されているアドレスである，

argv [i]+2

をatoi関数に渡すことによって取り出すことができるのです。atoi関数は，引数で与えられる文字列(へのポインタ）を整数値とみなして，その数値を戻り値とするライブラリ関数です。まさに，このような場合で使用されることを目的としている関数と思ってよいでしょう。

リスト1のプログラムは，タブの停止位置をコマンドラインから取り出してTABPOSという変数に代入しておき，標準入力から取り込んだ1行を1文字ずつ別の配列にコピーする過程でタブ文字('¥t')があれば，タブ文字の代わりに，

TABPOS-(コピーした文字数 % TABPOS)

だけの空白をコピーするというものです。プログラム自身は，この連載で標準入出力を説明した回の1行単位の処理を行うプログラムですから，やっていることはわかりますね。

ところで，リスト1のプログラムではコマンドの引数にタブの停止位置が指定されない（-tなどで始まる文字列がない）場合は，タブの停止位置を8として処理するようになっています(TABPOSの初期値が8なので)。引数にタブの停止位置がないとき（argcが1のとき）はコマンドの使用法を画面にプリントしてプログラムの処理を停止するというプログラムも考えられます。このときはargvの解釈を始める前に，

```
if(argc==1) {
  printf("usage:expand −tタブの位置 ¥n");
  printf("入力：標準入力，出力：標準出力¥n");
  exit(1);
}
```

などというif文を挿入しておけばよいでしょう（本来ならこのようなメッセージは標準エラー出力に書かれるものですが，ここでは標準出力に書いています)。この場合はコマンドが正常終了したわけではないのでexit関数の引数に0以外を指定していますが，コマンドが正常終了した場合にmain関数は値を返さないので，このexit関数の引数にはほとんど意味がありません。

### ●関数へのポインタを利用するプログラム

リスト2は台形公式を使って引数で与えられる関数の数値積分(定積分)を行うプログラムです。Integralという関数が数値積分を行う関数で，積分区間の下限(from)，上限(to)，繰り返し回数(loop)，積分する関数へのポインタ（func）を引数で指定するようになっています。

台形公式の原理は単純です。定積分とは与えられた積分区間でｘ軸と関数の囲む部分の面積を求めることです（関数のグラフを思い浮かべてください)。この積分区間を適当な数に分割し，各区間の占める面積を台形の面積

**リスト2　関数へのポインタを利用するプログラム**

```
 1: /*
 2:         関数へのポインタを使用するプログラム例
 3:
 4:         (台形公式によって関数の数値積分を行う)
 5: */
 6: #include <math.h>
 7:
 8: double Integral(from,to,loop,func)
 9: double from;
10: double to;
11: int    loop;
12: double (*func)();
13: {
14:     double step=(to-from)/loop;
15:     double y0,y1,area=0;
16:
17:     y0=(*func)(from);
18:     while(from<to){
19:         from+=step;
20:         y1=(*func)(from);
21:         area+=(y0+y1)*step/2;/* 台形の面積 */
22:         y0=y1;
23:     }
24:     return (area);
25: }
26:
27: double F1(x)        /* 被積分関数 その1 */
28: double x;
29: {
30:     return (sqrt(1-x*x));
31: }
32:
33: double F2(x)        /* 被積分関数 その2 */
34: double x;
35: {
36:     return (1/(1+x*x));
37: }
38:
39: main()
40: {
41:     printf("∫√(1-x*x)dx=%f¥n",
42:         Integral(0.0,1.0,2000,(double (*)())F1));
43:
44:     printf("∫1/(1+x*x)dx=%f¥n",
45:         Integral(0.0,10.0,2000,(double (*)())F2));
46: }
```

**リスト2の実行結果**

```
∫√(1-x*x)dx=0.785395
∫1/(1+x*x)dx=1.471128
```

で近似して，それを合計するのが台形公式です。これ以上プログラムのアルゴリズムの説明は不要だと思います。

リスト2ではXCのVer.1でコンパイルする(まだそんな人がいるのかなあ) ことも考慮して，関数のポインタによる関数の呼び出しは＊と ( ) を使用する古い形式で行っています。このほかに，関数へのポインタを仮引数で宣言する方法，関数へのポインタを引数に持つ関数を呼び出す方法に注意してプログラムを眺めてみてください。また，ANSI Cでの新しい形式に書き直してみるのもよいでしょう。

なお，プログラムの先頭の，

```
#include <math.h>
```

は，プログラム中で使用するsqrt関数(平方根を求める関数) の戻り値を宣言してあるmath.hというファイルを取り込むことを意味しています。sinやcosといった数値演算用の関数を使用するときのおまじないと思ってもよいでしょう。リスト2ではsqrt関数しか使用していないので，この1行の代わりに，

```
double sqrt( );
```

という1行を書いても同じことです。

## おわりに

本当はあと1回ぐらいポインタの話をしようと思っていたのですが，説明をしているうちに気が変わってしまったので，ポインタの話は今回で一応終わりにします(あとでもう1回出てくる予定ですが)。前回の説明で基礎知識は十分だと思ったのがその理由です。ポインタはC言語で最大の難関といわれています。しかし，実際のプログラムではごく限られた使用法しかされてないのが実情です。特に今回説明した関数へのポインタなどは特定の分野以外ではまず必要ありません (関数のポインタは配列と対比して考えると興味深いのであえて取り上げてみました)。

極論すればポインタなんか使わなくてもC言語のプログラムは書けるのです。C言語の初心者の多くの人は，プログラムで使うか使わないかわからないのに，広範囲な概念を持つポインタの一般論を聞いてくじけてしまっているのではないでしょうか。

たしかに，C言語ではポインタが必要な場面がいくつかあります。前回説明した，参照呼び出しを(疑似的に)実現するために関数の引数で使用する場合もそのひとつですが，これはポインタの概念をすべて知っていなくても理解することはできます。

このように必要なところから少しずつ理解していくようにすれば，ポインタというものは自然と理解できるようになるのではないでしょうか。

さて，来月は構造体の説明をします。構造体はC言語では配列や文字列に匹敵する非常に大事なデータ構造です。構造体を使いこなせるようになればC言語の学習も終わりに差しかかったと思ってよいでしょう。さあ，そろそろラストスパートですね。それでは来月またお会いしましょう。

### ◆基礎力を高めようの解答

**設問1**

宣言の意味は次の通り。

1) pはint型の1次元配列であることを示す。要素数は不明。配列要素のための領域は確保されない。

2) qはint型データを指し示すポインタ変数であることを示す。アドレス値を格納するための領域が確保される。初期値は与えられていない。

3) rはint型の1次元配列であることを示す。要素数は1。配列要素のための領域が確保され，要素が0に初期化される。

4) sはint型データを指し示すポインタ変数であることを示す。アドレス値を格納するための領域が確保され，0 (アドレス値が0) に初期化されている。

**解説**

1)の宣言だけ変数のための領域を確保しないことに注意。4)の初期化は {} で囲まないのが普通だが，このような宣言も許される。

**設問2**

宣言の意味は次の通り。

1) fpaは1次元配列名である。要素数は不明。各要素はint型を戻り値とする関数へのポインタである。配列要素のための領域は確保されない。

2) fpbは1次元配列名である。要素数は3である。各要素はint型を戻り値とする関数へのポインタである。配列要素のための領域が確保される。

3) fpcは関数名である。関数の戻り値がint型データへのポインタを要素とする配列へのポインタであることを示す。配列の要素数は不明である。配列要素のための領域は確保されない。また，関数の本体は定義されていない。

4) fpdは関数名である。関数の戻り値がint型データへのポインタを要素とする配列へのポインタであることを示す。配列の要素数は3である。配列要素のための領域は確保されない。また，関数の本体は定義されていない。

**解説**

1)と2)は配列の宣言，3)と4)は関数の (戻り値の) 宣言である。3)と4)は関数の戻り値がポインタへのポインタであることを宣言しているだけなので，4)の配列の要素数の宣言には意味がない。3)と4)はまったく同じ宣言と考えることもできる。

この設問のような複雑な宣言は，名前の近くに () があるか [] があるかを見ればわかりやすい。たとえば，

```
int (*fpa [ ])( );
```

はfpaの近くは [ ] であるから1次元配列 (の要素のデータ型) の宣言である。fpa [ ] をXと置き換えてみると，

```
int (*X)( );
```

であるから，Xは関数へのポインタであることがわかる。さらにXの部分はfpa[ ]だからこれは1次元配列である。したがって，その要素が関数へのポインタである1次元配列を宣言しているのがわかる。逆に，

```
int (*fpc( )) [ ];
```

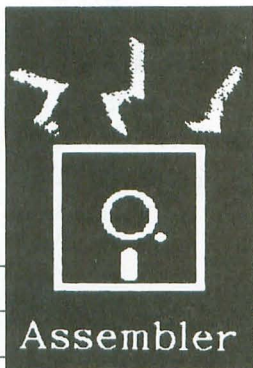
ではfpcの近くが ( ) であるから関数 (の戻り値のデータ型) の宣言であることがわかる。

ところで，1)〜4)はどれも配列要素あるいは関数の戻り値データ型を宣言しているだけであるが，1)と2)は配列の初期化を，3)と4)は関数の本体の定義を続けて記述することができる。たとえば，1)と3)を例に取ると，

```
int (*fpa [ ])( ) = {F1, F2, F3};
```

とか，

```
int (*fpc(x)) [ ]
int x;
{
    .........
    return … ;
}
```

といった宣言もできる (F1，F2，F3はint型を戻り値とする関数名とする)。

# シードフィルによる塗り潰し

Murata Toshiyuki 村田 敏幸

今回はシードフィルを使用したペイントルーチンの作成です。基本的な考え方は，シードピクセルを順に探し，塗り潰していくというもの。しかし，真正直にやっていたのでは遅くなるのが目に見えています。さて，村田氏はどのように対処するのでしょう。

今回のテーマはペイント，シードフィル（seed fill）による領域の塗り潰しだ。IOCSと同機能のペイントルーチンを作成してみる。

## シードフィルの基本

シードフィルの基本的な考え方は素朴なものだ。

1)　塗り潰しの開始点として指定された1ピクセル（シードと呼ぶ）を塗る

2)　いま塗ったシードピクセルの周囲から塗り潰すべきピクセルを探す

3)　そのピクセルを新たなシードとして1)，2)同様の処理を繰り返す

4)　新たなシードとなるピクセルがなくなった時点で塗り潰しは完了している

ここで，まっさらの画面を中央から塗り潰す場合を例にとるまでもなく，2)のステップで見つかるシードの候補は1点だけとは限らない。見つけたシード候補はあとでまな板に乗せるべく，すべてどこかに記憶しておく必要がある。そのあたりを考慮して，アルゴリズムをいくらか具体化するとつぎのようになる。

1')　シードピクセルを塗る

2')　いま塗ったシードピクセルの周囲から塗り潰されるべきピクセルを探し，シードの候補としてその座標をバッファに記憶する

3')　バッファから1点取り出し，“そのピクセルがまだシードとしての資格を持っている”なら，新たなシードとして1)，2)同様の処理を繰り返す

4')　バッファが空になった時点で塗り潰しは完了している

新たなシードを決める際には，バッファ中のどのピクセルを選んでもよいが，新しい順か，古い順かのどちらかに統一するのがふつうだろう。バッファをスタック構造にすれば自然に新しい順，キューにすれば古い順になる[1]。また，取り出し順序によらず，バッファから取り出したピクセルは無条件にシードとせずに，いま一度シードとしての資格審査にかける必要がある。同じピクセルがダブってバッファに格納される場合があり，バッファに格納した段階では未処理のピクセルであったものが取り出したときにはすでに処理済みになっている（もう塗り潰されている）可能性があるからだ。

さて，左で示した単純なシードフィルのアルゴリズムではピクセル単位で処理しているがために，塗り潰す範囲が少し広くなるとバッファに大量のメモリを必要とするようになる。また，同じピクセルを重複してバッファに格納する率も高く，その重複分がさらにバッファを食い潰し，効率も低下させている。

そこで，現実のペイントルーチンではもっとまとまった単位，水平線分単位で処理するのがふつうだ。塗り潰すべきピクセルが横方向に並んでいたらそれらをひとまとめにして，バッファには代表のピクセルひとつだけを格納する。その具体的な手順をつぎに示す。図1も併せて見てほしい（図の例ではバッファにキューを採用している）。

1)　シードピクセル(x, y)から左右方向に走査し，塗り潰しの境界を探す（得られた境界のx座標をそれぞれ$x_L$, $x_R$とおく）

2)　線分($x_L$, y)−($x_R$, y)を描く

3)　2)で描いた線分のすぐ上，線分($x_L$, y−1)−($x_R$, y−1)を左から走査して，“塗り潰すべきピクセルが横に連続している部分”をすべて探し，それぞれのもっとも右側の座標をバッファに格納する（ただし，境界が見つからないうちに右端に達した場合は，その右端の座標をバッファに格納する）

4)　2)で描いた線分のすぐ下，線分($x_L$, y+1)−($x_R$, y+1)に対して3)同様の処理を行う

5)　以下，バッファから有効なシードを取り出して

1)　ちなみに，画面中央から塗り潰したときに，色が中央から広がっていくようならそのペイントルーチンはキュー方式，先にいずれかの画面の端に達してから残りに取りかかるようならスタック方式と判断できる。

は，1)～4)を繰り返す

6)　バッファが空になった時点で塗り潰しは完了している

アルゴリズムをこのように変更することにより，バッファの消費は大幅に抑えられる。また，余分なピクセルがバッファに格納されることもほとんどなくなり，とくに，領域に穴が空いていない場合には重複は一切生じない。

2) 一般に，キュー方式のほうがバッファの消費量が少ないことが知られている。

## とりあえずペイントルーチンを作る

リスト1にペイントルーチンの第1版を示す。効率にはこだわらず，素直にアルゴリズムどおりに作ってみた。シードの候補を溜め込むバッファにはキュー(実装上はリングバッファ)を採用している[2]。

このサブルーチンgpaintにはIOCSコールの

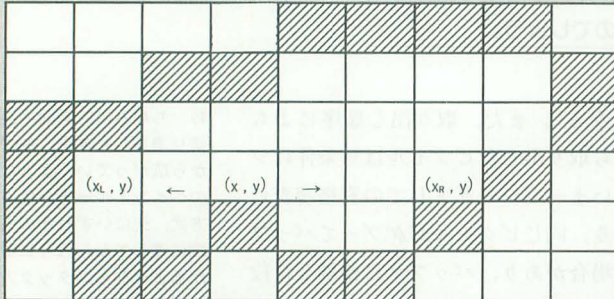
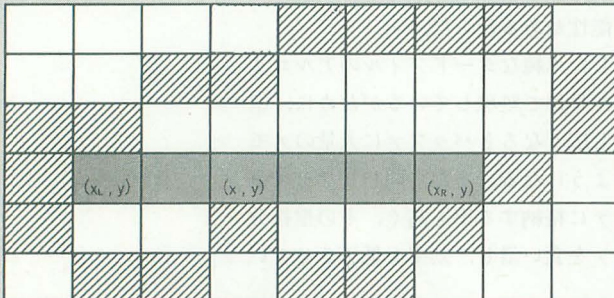
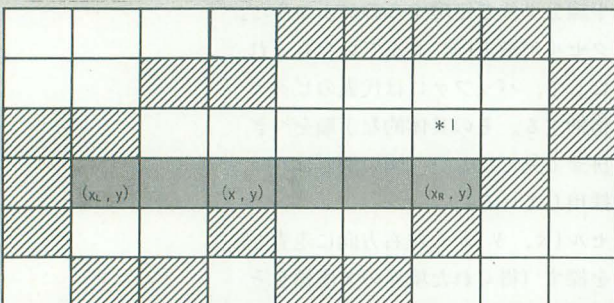
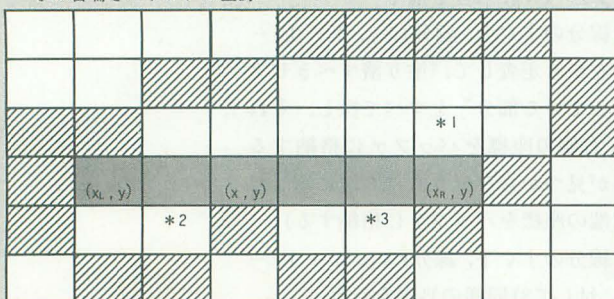
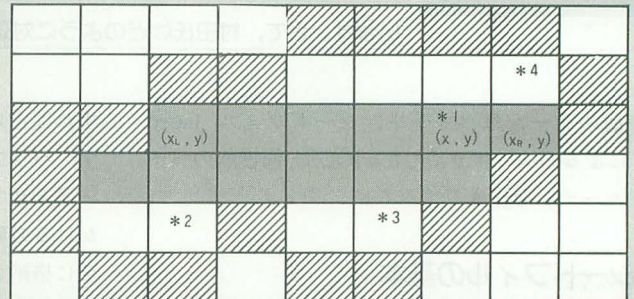
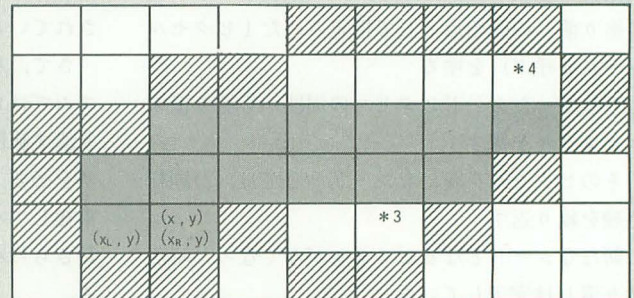
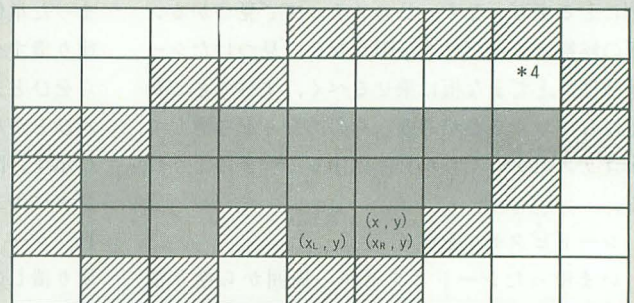
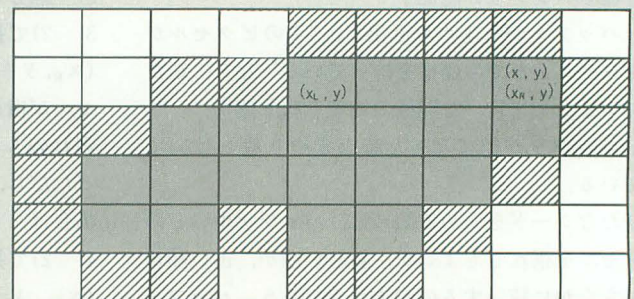



図1　スキャンラインシードフィルアルゴリズム

PAINTと同じ構造で引数列を用意し，その先頭アドレスをスタックに積んで渡す。引数列の具体的な構造はリスト中の10～16行に示してある。シードの候補を溜め込むバッファは，塗り潰す領域の形の複雑さに応じて呼び出し側で必要な大きさだけ用意し，gpaintには先頭アドレスと最終アドレス＋1の形で渡す。バッファには数Kバイトもあれば十分だ。

gpaintが塗り潰す領域は，上下左右に隣接する（4連接）同じ色のピクセル群として定義されている。ペイントルーチンには，適当な境界色で囲まれた閉じた領域を塗り潰す流儀や，上下左右だけではなく斜め方向に隣接するピクセルも塗り潰す（8連接）流儀もあるが，ここではIOCSの仕様に準じた。

では，プログラムを順に見ていこう。真っ先に46～57行で初期シードがクリッピングウィンドウ内にあるかどうかを調べている。ウィンドウ内であれば，そのG-RAMアドレスをa0に求め（59行），そのピクセルの色をd6に拾っておく（61行）。この色が塗り潰すべき領域の色となる。続いて，描画色をd7に取り出し，領域色と比較する（63～64行）。両者が一致するようなら，初期シードピクセルがすでに描画色になっているわけだから，何もせずにサブルーチンから戻る。

67～80行はキュー周りの初期化だ。最初の2行で読み出し位置を表すポインタa3と書き込み位置を表すポインタa4が，ともにバッファ先頭を指すようにし，キューを空にしている。話が前後するが，メインの処理を簡単にするために，このキューにはシード候補の座標と同時にG-RAMアドレスも格納するようにしてある。x，y座標に各2バイト，G-RAMアドレスに4バイトだから，1ピクセルあたりのバッファの消費量は8バイトとなる。

ここで，リングバッファに対してデータを読み書きするときには，読み書きポインタがバッファ末尾に達したかどうかを調べ，末尾に達したら先頭を指すよう修正する必要があることを思い出してほしい。この都合上，gpaintに渡されるバッファの大きさはきっかり8の倍数であることが望ましく，71～77行では念のため，バッファの大きさが変な値になっていないかどうか調べ，バッファサイズが8の倍数になるようバッファ最終アドレスを調整している。

84行からがメインループだ。最初の1回だけはすでにレジスタにシードの座標とアドレスが格納されているので，ループの途中，92行に飛び込む。2度目以降は84～90行でキューからシードを取り出す処理がその直前に入る。

109～117行でシードピクセルから右方向に走査して領域色から非領域色への変わり目，あるいは，クリッピングウィンドウの右端を探す。その位置が，いま塗り潰す水平線分の右端（$x_R$）となる。領域色かどうかの判定に先立ってx座標を増加させている関係で，111～115行のループを抜けたときのd2は，求めるx座標より1大きい。117行はこの分の補正だ。120～129行では同様にシードピクセルから左に走査して，水平線分の左端x座標（$x_L$）をd0に求めている。このとき，同時に対応するG-RAMアドレスがa0に求まる。

ここまでで得た結果を使い，131～136行で水平線分を描く。それから，142～146行で真上，149～153行で真下の1ラインからつぎのシードの候補となるピクセルを探し，キューに追加する。シードの候補を探す処理は166行以下のサブルーチンseapixにまとめてある。seapixでは，ラインの左端から非領域色ピクセルを飛ばして領域色ピクセル列の左端を探し（169～171行），そこから非領域色ピクセルにぶつかるまでポインタとx座標を進める（175～180行）という手順で，領域色ピクセル列の右端座標とG-RAMアドレスを求めている。得られた座標とアドレスは182～190行でキューに追加する。

169～171行や175～177行のループでは走査があらかじめ決められた走査範囲（x座標が$x_L$～$x_R$）を越えて行われないようにするために，dbeq，dbneが微妙な使われ方をしている。dbeq，dbneのループをさらに169～193行のループで括ってある関係で，少し気を抜くと，走査の進み具合とループカウンタとのつじつまが合わなくなってしまうのだ。172行，178行で，ループを抜けた直後にしている条件判断や，外側のループ最後の192行でループカウンタd3が－1になっていないかどうか調べ直している意味をよく考えてみてほしい。

上下のラインの走査が済んだ時点でひとつのシードピクセル（で代表される水平線分）の処理が完了する。以下，同様の処理をキューが空になるまで繰り返す。キューが空かどうかは読み出し位置と書き込み位置が一致しているかどうかで判断している（155行）。

## ペイントルーチンの高速化を目指す

リスト1は真正直に作ってあるだけあって，実行速度はかなり遅い。平均的な性能はIOCSコールのPAINTよりも数段落ちる。IOCS.X版はおろかROM版にも及ばない。そこで，ぼちぼちと高速化に取りかかる。せっかく作るのだから，せめてIOCSよりは速くしてやりたい。そこで，定石どおり，アルゴリズムから見直してみる。図2のような図形を点Pから塗り潰す場合を例に，アルゴリズムのおさらいついで，処理の流れを追い，無駄がないかどうか

検討してみよう。

まず，シードPから左右方向に境界を探すとA，Bが得られる。AB間を塗り潰したら，ABの上下のラインであるCD，EF中から領域色の部分を探す。CDはまるまんま領域色，EF中ではGH間が領域色だから，それぞれの代表として右端のD，Hをキューに追加する。これで1ライン分の処理が済んだので，キューから新たなシードを取り出す。リスト1の処理順序に従えば，点Dが取り出される。

Dからの左右方向の境界検索時，左方向へはDからIまで，右方向へはDからJまでの範囲を走査することになる。ここでひとつ無駄な処理が見つかった。CD間はシードがPのときに一度走査済みであり，すべて領域色だということがわかっている。この情報を利用すればCD間の再走査は不要となり，左方向の走査はCの左のピクセルから始めるよう手抜きができる。もちろん，そのためにはバッファに記憶しておく情報量を増やしておかなければならない。つまり，上下のラインを走査したときに領域色が連続した部分が見つかったら，これまでのように右端だけをバッファに記録するのではなく，左端もセットにして覚えておくのだ。シードを点から線分に拡張するといってもよい。

IJを得，塗り潰したあとの上下のラインの走査にも同様の無駄がある。初期シードを除けば，上か下かに“そのシードをバッファに登録した親のライン”があるはずだ。親のラインはすでに一度走査されているから，親ラインとの重複部分は再走査の必要がない。いまの例でいうと，線分IJの下側走査範囲KLのうちCDの真下にあたる部分ABの走査が，ばっさり省略できる。この場合，走査範囲KLは2分されることに注意しよう。また，実際のプログラムでは親が上にあるか下にあるか判断できなければならないから，シードをバッファに登録する際に，親ラインのy座標という情報を付け加えることが必要になる。

以上の改良により，G-RAMを読み込む回数はだいたい1/3になる。線分描画時の書き込みと合わせて考えても，G-RAMアクセス回数は半分で済む計算だ。ある程度以上の面積を塗り潰す場合であれば，これはそのまま実行時間の半減につながる。

リスト2が改良版アルゴリズムを使ったペイントルーチンだ。細部も適当に最適化してある。たとえば，左右方向の境界検索を行う135～136行や146～147行のループはリスト1の111～115行，122～126行よりもずっとシンプルになっている。第1にループの中で座標をカウントアップしていた部分がなくなった。座標の変化は，ループに入る前と抜けたあとのG-RAMを指すポインタの差から逆算するようにしたのだ。また，ループの中でウィンドウ端に達したかチェックしていた部分もなくなった。ループに入る前にウィンドウ端までのピクセル数を計算してdbneを使うようにしたからだ。

毎回書いているような気もするが，何度も繰り返し実行されるループの中身を軽くするのは最適化の基本だ。その結果としてループの外の命令数が増えたとしても，実行回数を乗じた実行時間を考えれば十分おつりがくる。

では，アルゴリズムの改良に伴う変更部分をざっと拾っていこう。まず，キューに格納するシード候補についての情報が前述のように増えている（26～34行）。シード候補ひとつあたり，“シード線分”の両端点のG-RAMアドレスと座標（ただし，y座標は共用），それに“親ライン”のy座標の16バイトだ。この変更はキューに対する読み書きを行う部分のほかに，リングバッファの最終アドレスを決める部分（84行）にも影響している。

また，コーディング上の問題だが，情報量が増えたことにより，キューから取り出したシードの情報をレジスタに留めておくのが難しくなったため，キューからの読み出しを数カ所に分散せざるをえなくなった。メインループの先頭，キューから新たなシードの情報を取り出す時点では，すぐに必要になる両端点の座標とG-RAMアドレスだけをレジスタに転送している。

さらにみっともないことに，本来16バイト単位で進むはずの読み込みポインタは14バイトだけ進んだ中途半端な状態で一時停止する格好になる。しかも，両端点のx座標はあとでまた必要になったときに，再度メモリから読み込んでいたりもする。このため，初期シードもいったんキューに登録し，メモリ上に置いておくようにしなければならなくなった（92～97行）。1カ所の変更が連鎖的に変更を引き起こす悪い見本だ。

シードから左右方向へ塗り潰しの境界を探す131～149行は，走査範囲が重複しないよう修正されている。最適化による変更に埋もれて目立たなくなってはい

図2

るが，アルゴリズム上は重要な修正のひとつだ。

そして，もっとも大きく変更されたのが，シードの上下ラインを走査するサブルーチンseapixだ。親ラインとの重複部分を飛ばす処理（206〜227行）が付け加えられ，また，領域色ピクセル列の両端と現在処理中のシードのY座標をキューに登録するようになった（241〜245行，258〜262行）。

リスト2はリスト1に比べ，平均して2倍程度は速くなった。すでにROM版のPAINTの性能は優に越えている。が，IOCS.X版にはまだ及ばない。しかたなく最後の切札，水平線分描画のループ展開を施す。ここまでやってようやくIOCS.X版のPAINTの速度を（平均5〜10%程度ながら）越えることができる。リスト2に対する変更点のみリスト3に示す。

リスト3で使っている方法は，以前ボックスフィルやスキャンコンバージョンのときにやってみせた露骨なループ展開ではなく，よりメモリを食わず，それでいて平均的な速度が速い。本当はもっと早い時期に紹介するつもりだったのだが，ボックスフィルのときに示しそこねたらタイミングを逸してしまったのだ。

68000にはメモリに対して64バイトを1命令で書き込む手段がある。つぎの命令がそうだ。

```
movem.l d0-d7/a0-a7,-(aX)
```

これをG-RAMに適用できれば，全レジスタの上位/下位ワードにパレットコードを入れておくことで32ピクセルの書き込みが同時に行える計算になる。この命令を32個並べれば，最大1024ピクセルの水平線分が描けるわけだ。しかし，実際にはスタックポインタに適当な値を入れるわけにはいかないし，アドレスレジスタの1本はG-RAMへのポインタに使うので，リスト3では，

```
movem.l d5-d7/a2-a6,-(a1)
```

による16ピクセル同時書き込みを64個並べている。そのわりにソースが短いのは，276〜285行のマクロで逃げたからだ。

リスト3では，この64個並んだmovemの途中，適当な位置に飛び込むことで16ピクセル単位の任意の長さの水平線分を描く。16ピクセル未満の端数については，dbraループで処理している。また，ループ展開の効果が出ない最初から16ピクセル未満の場合は323〜324行の専用ループで処理し，時間のロスを軽減している。

## 動作試験

最後に動作試験用のプログラムをリスト4に示して終わろう。

リスト4は，図3のようなパターンで覆われた画面を中央からペイントするだけのプログラムだ。図3のパターンは，かなりペイントルーチンに負担をかける（バッファを大量に必要とする）ように作ってある。

また，リスト4は純粋に動作試験用プログラムであり，速度試験用ではないことを断っておく必要があるだろう。リスト4で速度試験をすると，単純さが幸いしてリスト1の版がIOCSを含むほかのどれよりも高速だという楽しい結果が出る。速度比較がしたかったら，もっと一般的な図形で試してみてほしい。

ところで，今回は処理速度に限ってIOCSと張り合ってみたわけだが，実はIOCSのPAINTには“バッファサイズがあまりなくても，結構複雑な領域を塗り潰すことができる”という，gpaintにはない長所がある。

リスト4の46〜48行を殺し，50〜51行を復活して，75行のバッファサイズを数100バイトにまで小さくしてみると，IOCSが頑張る様子を見ることができる。

どうやら，IOCSはバッファが一杯になったときに，少ないダメージで処理が継続できるよう細工しているらしい。たぶん，バッファが一杯になったら，古いデータからひとつずつ捨てているのだろう。シードの候補をひとつでも捨ててしまえば，塗り残しの可能性は生まれるわけだが，バッファが一杯になるほどの複雑さをもった穴だらけの領域であれば，反対側からの回り込みによって，たいていはカバーできる。

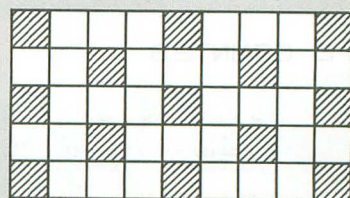
この点，gpaintは実に手抜きで，リスト1，2ともに，キューにシード候補を追加する際のバッファの溢れを判定していない。本来は，書き込み位置がぐるっと回って読み出し位置に追いつき，バッファが一杯になったら，せめてエラー終了するべきなのだが，そのチェックすらしていないのだ。

ただし，危険はない。書き込み位置が読み出し位置に追いついた場合は，メインループ最後のループ終了判定で“キューが空”と判断されてループを抜ける。また，書き込み位置が読み出し位置を追い越したりしたときには，バッファ一杯分のシード候補を捨てた形で処理がもうしばらく継続する。

IOCS相当の細工を施すのはそれほど難しくはないから，気が向いたら試してみてもらいたい。

＊

というところでペイントの話は終わる。次回は円の描画を取り上げる予定だ。

図3

## リスト1 GPAINT1.S

```
  1: *          ペイント
  2:
  3:            .include        gconst.h
  4:            .include        gmacro.h
  5: *
  6:            .xdef   gpaint
  7:            .xref   gramadr
  8:            .xref   cliprect
  9: *
 10:            .offset 0       *gpaintの引数構造
 11: *
 12: X0:        .ds.w   1       *初期シード座標
 13: Y0:        .ds.w   1       *
 14: COL:       .ds.w   1       *描画色
 15: TEMP:      .ds.l   1       *作業用領域先頭
 16: TEMPED:    .ds.l   1       *作業用領域末尾+1
 17: *
 18:            .offset -8      *スタックフレーム
 19: WORKSIZ:
 20: BUFTOP:    .ds.l   1
 21: BUFEND:    .ds.l   1
 22: _a6:       .ds.l   1
 23: _pc:       .ds.l   1
 24: ARGPTR:    .ds.l   1
 25: *
 26:            .offset 0       *シード候補
 27: *
 28: ADR:       .ds.l   1
 29: X:         .ds.w   1
 30: Y:         .ds.w   1
 31: *
 32:            .offset 0       *クリッピング領域
 33: *
 34: MINX:      .ds.w   1
 35: MINY:      .ds.w   1
 36: MAXX:      .ds.w   1
 37: MAXY:      .ds.w   1
 38: *
 39:            .text
 40:            .even
 41: *
 42: gpaint:
 43:            link    a6,#WORKSIZ
 44:            movem.l d0-d7/a0-a5,-(sp)
 45:
 46:            lea.l   cliprect,a5      *a5=クリッピング領域
 47:            movea.l ARGPTR(a6),a1    *a1=引数列
 48:
 49:            movem.w X0(a1),d0-d1     *(d0,d1)=初期シード
 50:            cmp.w   MINX(a5),d0      *初期シードが
 51:            blt     done             *  ウィンドウ外なら
 52:            cmp.w   MINY(a5),d1      *  何もせずに戻る
 53:            blt     done             *
 54:            cmp.w   MAXX(a5),d0      *
 55:            bgt     done             *
 56:            cmp.w   MAXY(a5),d1      *
 57:            bgt     done             *
 58:
 59:            jsr     gramadr          *a0=初期シードG-RAMアドレス
 60:
 61:            move.w  (a0),d6          *d6=領域色
 62:
 63:            move.w  COL(a1),d7       *d7=描画色
 64:            cmp.w   d6,d7            *描画色と領域色が等しいなら
 65:            beq     done             *  何もせずに戻る
 66:
 67:            movea.l TEMP(a1),a3      *a3=キュー内の読み出し位置
 68:            movea.l a3,a4            *a4=キュー内の書き込み位置
 69:                                     *(ともにバッファ先頭に初期化)
 70:
 71:            move.l  TEMPED(a1),d3    *d3=バッファ末尾+1
 72:            sub.l   a3,d3            *バッファ先頭<末尾ならば
 73:            bcs     done             *  何もせずに戻る
 74:            andi.l  #$ffff_fff0,d3   *d3=バッファサイズ(16の倍数)
 75:            beq     done             *バッファサイズが0ならば
 76:                                     *  何もせずに戻る
 77:            add.l   a3,d3            *d3=バッファ末尾+1
 78:
 79:            move.l  a3,BUFTOP(a6)    *バッファ先頭と末尾を
 80:            move.l  d3,BUFEND(a6)    *  ワークに格納しておく
 81:
 82:            bra     do               *メイン処理開始
 83:
 84: loop:      movea.l (a3)+,a0         *キューからシードを
 85:            move.w  (a3)+,d0         *  取り出す
 86:            move.w  (a3)+,d1         *
 87:
 88:            cmpa.l  BUFEND(a6),a3    *キューの読み出しポインタが
 89:            bcs     do               *  バッファ末尾に達したら
 90:            movea.l BUFTOP(a6),a3    *  先頭を指すよう修正する
 91:
 92: do:
 93: *          d0      シードx座標
 94: *          d1      シードy座標
 95: *          d6      領域色
 96: *          d7      描画色
 97: *          a0      シードG-RAMアドレス
 98: *          a3      キュー内のつぎの読み出し位置
 99: *          a4      キュー内のつぎの書き込み位置
100: *          a5      クリッピング領域
101:
102:            movea.l a0,a1            *a0=a1=シードのG-RAMアドレス
103:            move.w  d0,d2            *d0=d2=シードのx座標
104:
105:            cmp.w   (a1)+,d6         *すでに処理済みのシードならば
106:            bne     next             *  捨てる
107:
108:                                *右方向の境界を探す
109: rchk:      move.w  MAXX(a5),d3      *d3=ウィンドウ右端x座標
110:                                     *d2=x1
111: rloop:     cmp.w   d3,d2            *ウィンドウ右端に達したら
112:            bge     lchk             *  その直前の点が右の境界
113:            addq.w  #1,d2            *x1++
114:            cmp.w   (a1)+,d6         *(x1,y)が領域色のあいだ
115:            beq     rloop            *  繰り返す
116:
117:            subq.w  #1,d2            *d2=x1
118:
119:                                *左方向の境界を探す
120: lchk:      move.w  MINX(a5),d3      *d3=ウィンドウ左端x座標
121:                                     *d0=x0
122: lloop:     cmp.w   d3,d0            *ウィンドウ右端に達したら
123:            ble     filspn           *  その直前の点が左の境界
124:            subq.w  #1,d0            *x0--
125:            cmp.w   -(a0),d6         *(x0,y)が領域色のあいだ
126:            beq     lloop            *  繰り返す
127:
128:            addq.w  #1,d0            *d0=x0
129:            addq.l  #2,a0            *a0=(x0,y)のG-RAMアドレス
130:
131: filspn:    sub.w   d0,d2            *d2=線分のピクセル数-1
132:            movea.l a0,a2            *a0=a2=線分左端G-RAMアドレス
133:
134:            move.w  d2,d3            *線分(x0,y)-(x1,y)を描く
135: floop:     move.w  d7,(a0)+         *
136:            dbra    d3,floop         *
137:
138:            move.w  d0,d4            *d0=d4=x0(線分左端x座標)
139:            subq.w  #1,d4            *d4=x0-1
140:
141:                                *真上のスキャンラインを走査する
142: uchk:      subq.w  #1,d1            *y--
143:            cmp.w   MINY(a5),d1      *ウィンドウの上端を越えていたら
144:            blt     dchk             *  走査の必要はない
145:            lea.l   -GNBYTE(a2),a0   *a0=走査するライン左端
146:            bsr     seapix           *走査する
147:
148:                                *真下のスキャンラインを走査する
149: dchk:      addq.w  #1+1,d1          *y++
150:            cmp.w   MAXY(a5),d1      *ウィンドウの下端を越えていたら
151:            bgt     next             *  走査の必要はない
152:            lea.l   GNBYTE(a2),a0    *a0=走査するライン左端
153:            bsr     seapix           *走査する
154:
155: next:      cmpa.l  a3,a4            *キューが空になるまで
156:            bne     loop             *  繰り返す
157:
158: done:      movem.l (sp)+,d0-d7/a0-a5
159:            unlk    a6
160:            rts
161: *
162: *          線分(x0,y-1)-(x1,y-1)または
163: *          線分(x0,y+1)-(x1,y+1)を走査して
164: *          シードとなる点を探しキューに追加する
165: *
166: seapix:
167:            move.w  d4,d0            *d0=走査開始x座標-1
168:            move.w  d2,d3            *d3=線分のピクセル数-1
169: sealp1:    addq.w  #1,d0            *非領域色部分を飛ばす
170:            cmp.w   (a0)+,d6         *
171:            dbeq    d3,sealp1        *
172:            beq     seapx0           *領域色が見つかった
173:            rts                      *領域色は見つからなかった
174:
175: sealp2:    addq.w  #1,d0            *領域色部分を飛ばす
176:            cmp.w   (a0)+,d6         *
177: seapx0:    dbne    d3,sealp2        *
178:            beq     seapx1           *
179:            subq.l  #2,a0            *ポインタとx座標は
180:            subq.w  #1,d0            *  進み過ぎている
181:
182: seapx1:    subq.l  #2,a0            *a0=領域色部分の右端
183:            move.l  a0,(a4)+         *見つけたシード候補を
184:            move.w  d0,(a4)+         *  バッファに登録する
185:            move.w  d1,(a4)+         *
186:            addq.l  #2,a0            *つぎの走査に備える
187:
188:            cmpa.l  BUFEND(a6),a4    *キューの書き込みポインタが
189:            bcs     seanx            *  バッファ末尾に達したら
190:            movea.l BUFTOP(a6),a4    *  先頭を指すよう修正する
191:
192: seanx:     tst.w   d3               *走査範囲の右端に達するまで
193:            bpl     sealp1           *  繰り返す
194:
195:            rts
196:
197:            .end
```

## リスト2 GPAINT2.S

```
  1: *          ペイント(高速化版)
  2:
  3:            .include        gconst.h
  4:            .include        gmacro.h
  5: *
  6:            .xdef   gpaint
  7:            .xref   gramadr
  8:            .xref   cliprect
  9: *
 10:            .offset 0       *gpaintの引数構造
 11: *
 12: X0:        .ds.w   1       *初期シード座標
 13: Y0:        .ds.w   1       *
 14: COL:       .ds.w   1       *描画色
```

▶先日,「満開の電子ちゃん」というべきところをうっかり「電開の(ピ~)」と口走ってしまい,大恥をかいた。 深町 忠利(23)神奈川県

```
 15: TEMP:    .ds.l   1         *作業用領域先頭
 16: TEMPED:  .ds.l   1         *作業用領域末尾+1
 17: *
 18:          .offset -8        *スタックフレーム
 19: WORKSIZ:
 20: BUFTOP:  .ds.l   1         *リングバッファ先頭
 21: BUFEND:  .ds.l   1         *リングバッファ末尾+1
 22: _a6:     .ds.l   1
 23: _pc:     .ds.l   1
 24: ARGPTR:  .ds.l   1
 25: *
 26:          .offset 0         *シード候補
 27: *
 28: LADR:    .ds.l   1         *左端G-RAMアドレス
 29: LX:      .ds.w   1         *左端x座標
 30: Y:       .ds.w   1         *y座標
 31: RADR:    .ds.l   1         *右端G-RAMアドレス
 32: RX:      .ds.w   1         *右端x座標
 33: PREVY:   .ds.w   1         *親のy座標
 34: SEEDSIZ:
 35: *
 36:          .offset 0
 37: *
 38: MINX:    .ds.w   1
 39: MINY:    .ds.w   1
 40: MAXX:    .ds.w   1
 41: MAXY:    .ds.w   1
 42: *
 43:          .text
 44:          .even
 45: *
 46: gpaint:
 47:          link    a6,#WORKSIZ
 48:          movem.l d0-d7/a0-a5,-(sp)
 49:
 50:          lea.l   cliprect,a5       *a5=クリッピング領域
 51:          movea.l ARGPTR(a6),a1     *a1=引数列
 52:
 53:          movem.w X0(a1),d0-d1      *(d0,d1)=初期シード
 54:          cmp.w   MINX(a5),d0       *初期シードが
 55:          blt     done              *  ウィンドウ外なら
 56:          cmp.w   MINY(a5),d1       *  何もせずに戻る
 57:          blt     done              *
 58:          cmp.w   MAXX(a5),d0       *
 59:          bgt     done              *
 60:          cmp.w   MAXY(a5),d1       *
 61:          bgt     done              *
 62:
 63:          jsr     gramadr           *a0=初期シードG-RAMアドレス
 64:
 65:          move.w  (a0),d6           *d6=領域色
 66:
 67:          move.w  COL(a1),d7        *d7=描画色
 68:          cmp.w   d6,d7             *描画色と領域色が等しいなら
 69:          beq     done              *  何もせずに戻る
 70: *        swap.w  d7
 71: *        move.w  COL(a1),d7
 72:
 73:          movea.l TEMP(a1),a3       *a3=キュー内の読み出し位置
 74:          movea.l a3,a4             *a4=キュー内の書き込み位置
 75:                                    *(ともにバッファ先頭に初期化;
 76:
 77:          movea.l TEMP(a1),a3       *a3=キュー内の読み出し位置
 78:          movea.l a3,a4             *a4=キュー内の書き込み位置
 79:                                    *(ともにバッファ先頭に初期化;
 80:
 81:          move.l  TEMPED(a1),d3     *d3=バッファ末尾+1
 82:          sub.l   a3,d3             *バッファ先頭<末尾ならば
 83:          bcs     done              *  何もせずに戻る
 84:          andi.l  #$ffff_fff0,d3    *d3=バッファサイズ(16の倍数)
 85:          beq     done              *バッファサイズが0ならば
 86:                                    *  何もせずに戻る
 87:          add.l   a3,d3             *d3=バッファ末尾+1
 88:
 89:          move.l  a3,BUFTOP(a6)     *バッファ先頭と末尾を
 90:          move.l  d3,BUFEND(a6)     *  ワークに格納しておく
 91:
 92:          move.l  a0,(a4)+          *初期シードをキューに追加
 93:          move.w  d0,(a4)+          *
 94:          move.w  d1,(a4)+          *
 95:          move.l  a0,(a4)+          *
 96:          move.w  d0,(a4)+          *
 97:          move.w  d1,(a4)+          *
 98:
 99:          movea.l a0,a1             *初期シード特有の
100:          move.w  d0,d2             *  つじつま合わせ
101:          lea.l   PREVY(a3),a3      *
102:          cmpa.l  d3,a4             *
103:          bcs     do                *
104:          lea.l   -SEEDSIZ(a4),a4 *
105:          bra     do                *
106:
107: loop:    movea.l (a3)+,a0          *LADR
108:          move.w  (a3)+,d0          *LX
109:          move.w  (a3)+,d1          *Y
110:          movea.l (a3)+,a1          *RADR
111:          move.w  (a3)+,d2          *RX
112:
113: do:
114: *        d0      左側シードx座標
115: *        d1      左側シードy座標
116: *        d2      右側シードx座標
117: *        d6      非境界色(この色の点を塗り潰す)
118: *        d7      描画色
119: *        a0      左側シードG-RAMアドレス
120: *        a1      右側シードG-RAMアドレス
121: *        a3      キュー内のつぎの読み出し位置-2
122: *        a4      キュー内のつぎの書き込み位置
123: *        a5      クリッピング領域
124:
125:          movea.l a0,a2             *a0=a2=シード右端のG-RAMアドレス
126:
127:          cmp.w   (a1)+,d6          *すでに処理済みのシードならば
128:          bne     next              *  捨てる
129:
130:                          *右方向の境界を探す
131: rchk:    move.w  MAXX(a5),d3       *
132:          sub.w   d2,d3             *d3=走査する最大ピクセル数
133:          subq.w  #1,d3             *d3=0ならば
134:          bmi     lchk              *  走査は不要
135: rloop:   cmp.w   (a1)+,d6          *
136:          dbne    d3,rloop          *
137:          beq     lchk              *
138: *        subq.l  #2,a1             *
139:
140:                          *左方向の境界を探す
141: lchk:    move.w  MINX(a5),d3
142:          sub.w   d0,d3             *d3=-走査する最大ピクセル数
143:          neg.w   d3                *d3=走査する最大ピクセル数
144:          subq.w  #1,d3             *d3=0ならば
145:          bmi     filspn            *  走査は不要
146: lloop:   cmp.w   -(a0),d6          *
147:          dbne    d3,lloop          *
148:          beq     filspn            *
149:          addq.l  #2,a0             *
150:
151: filspn:  move.l  a2,d3             *d3=走査開始前の位置
152:          sub.l   a0,d3             *d3=左へ走査したバイト数
153:          lsr.w   #1,d3             *d3=左へ走査したピクセル数
154:          sub.w   d3,d0             *d0=線分左端x座標
155:
156:          move.l  a1,d2             *d2=走査終了後の右端位置
157:          sub.l   a0,d2             *d2=左端から右端のバイト数
158:          lsr.w   #1,d2             *d2=線分のピクセル数
159:          subq.w  #1,d2             *d2=線分のピクセル数-1
160:
161:          movea.l a0,a2             *a0=a2=線分左端G-RAMアドレス
162:
163:          move.w  d2,d3             *線分描画
164: floop:   move.w  d7,(a0)+          *
165:          dbra    d3,floop          *
166:
167:                          *真上のスキャンラインを走査する
168: uchk:    add.w   d0,d2             *d2=線分右端x座標
169:
170:          move.w  d1,d4             *d4=y
171:          subq.w  #1,d1             *y--
172:          cmp.w   MINY(a5),d1       *ウィンドウの上端を越えていたら
173:          blt     dchk              *  走査の必要はない
174:          lea.l   -GNBYTE(a2),a0    *a0=走査するライン左端
175:          bsr     seapix            *走査する
176:
177:                          *真下のスキャンラインを走査する
178: dchk:    addq.w  #1+1,d1           *y++
179:          cmp.w   MAXY(a5),d1       *ウィンドウの下端を越えていたら
180:          bgt     next              *  走査の必要はない
181:          lea.l   GNBYTE(a2),a0     *a0=走査するライン左端
182:          bsr     seapix            *走査する
183:
184: next:    addq.l  #2,a3             *読み出しポインタを
185:                                    *  次のデータに進める
186:          cmpa.l  BUFEND(a6),a3     *
187:          bcs     nottop            *
188:          movea.l BUFTOP(a6),a3     *
189:
190: nottop:  cmpa.l  a3,a4             *キューが空になるまで
191:          bne     loop              *  繰り返す
192:
193: done:    movem.l (sp)+,d0-d7/a0-a5
194:          unlk    a6
195:          rts
196: *
197: *        上下ラインから
198: *        シードとなる領域色部分を探しキューに追加する
199: *
200: seapix:
201:          movea.l a0,a1             *a0=a1=走査開始位置
202:          suba.w  d0,a1             *
203:          suba.w  d0,a1             *a1=走査開始ラインの
204:                                    *  物理的な左端(x=0)のアドレス
205:
206:          cmp.w   (a3),d1           *走査するラインは親ライン?
207:          bne     seapx1            *  違うのならばふつうに走査する
208:
209:          move.w  LX-PREVY(a3),d3 *d3=走査範囲のうち
210:          sub.w   d0,d3             *  親ライン左端からのはみ出し分
211:          ble     seapx0            *はみ出しはなかった
212:          subq.w  #1,d3             *d3=ループカウンタ
213:
214:          move.l  a0,-(sp)          *走査範囲のうち
215:          bsr     seapx2            *  親ライン左端より
216:          movea.l (sp)+,a0          *  左の部分を走査
217:
218: seapx0:
219:          move.w  d2,d3             *d3=走査範囲のうち
220:          move.w  RX-PREVY(a3),d5 *
221:          sub.w   d5,d3             *  親ライン右端からのはみ出し分
222:          ble     seartn            *はみ出しはなかった
223:          subq.w  #1,d3             *d3=ループカウンタ
224:
225:          add.w   d5,d5             *a0=
226:          lea.l   2(a1,d5),a0       *  走査開始位置
227:          bra     seapx2            *走査範囲のうち
228:                                    *  親ライン右端より
229:                                    *  右の部分を走査
230:
231: seapx1:  move.w  d2,d3             *
232:          sub.w   d0,d3             *d3=走査範囲のピクセル数-1
233: seapx2:
234: sealp1:  cmp.w   (a0)+,d6          *非領域色部分を飛ばす
235:          dbeq    d3,sealp1         *
236:          bne     seartn            *
237:
238:          move.l  a0,d5
239:          subq.l  #2,d5             *d5=領域色部分右端アドレス
240:
241:          move.l  d5,(a4)+          *LADR
242:          sub.l   a1,d5             *
243:          lsr.w   #1,d5             *
244:          move.w  d5,(a4)+          *LX
245:          move.w  d1,(a4)+          *Y
246:
247:          subq.w  #1,d3             *
248:          bmi     bound             *走査範囲右端に達した
```

▶おいらが初めてやったモニタの仕事が載ってるなあ。ウフフ。にったらニッタラしながら何度も読み返しましたよ。でも，安井嬢の半分……まだまだよのう。

弦元　達也(20)香川県

```
249:
250: sealp2: cmp.w    (a0)+,d6          *領域色部分を飛ばす
251:         dbne     d3,sealp2         *
252:         beq      bound             *
253:         subq.l   #2,a0             *
254:
255: bound:  move.l   a0,d5
256:         subq.l   #2,d5             *d5=領域色部分左端アドレス
257:
258:         move.l   d5,(a4)+          *RADR
259:         sub.l    a1,d5             *
260:         lsr.w    #1,d5             *
261:         move.w   d5,(a4)+          *LX
262:         move.w   d4,(a4)+          *PREVY
263:
264:         cmpa.l   BUFEND(a6),a4     *キューの書き込みポインタが
265:         bcs      seanx             *  バッファ末尾に達したら
266:         movea.l  BUFTOP(a6),a4     *  先頭を指すよう修正する
267:
268: seanx:  tst.w    d3                *走査範囲の右端に達するまで
269:         bpl      sealp1            *  繰り返す
270:
271: seartn: rts
272:
273:         .end
```

## リスト3 GPAINT3.S

```
  1: *             ペイント（最終版）
 70:               swap.w  d7
 71:               move.w  COL(a1),d7
138:               subq.l  #2,a1                *
163: *
164:               bsr     hline                *水平線分描画
165: *
273: *
274: *             水平線分描画
275: *
276: H128          macro
277:               movem.l d5-d7/a2-a6,-(a1)
278:               movem.l d5-d7/a2-a6,-(a1)
279:               movem.l d5-d7/a2-a6,-(a1)
280:               movem.l d5-d7/a2-a6,-(a1)
281:               movem.l d5-d7/a2-a6,-(a1)
282:               movem.l d5-d7/a2-a6,-(a1)
283:               movem.l d5-d7/a2-a6,-(a1)
284:               movem.l d5-d7/a2-a6,-(a1)
285:               endm
286: *
287:               H128     *1024
288:               H128     * :
289:               H128     * :
290:               H128     * :
291:               H128     *512
292:               H128     *384
293:               H128     *256
294:               H128     *128
295: hlin00:       movem.l (sp)+,d5-d6/a2-a6
296:               rts
297: *
298: hline:
299:               move.w  d2,d3
300:               addq.w  #1,d3
301:               move.w  d3,d4
302:               andi.w  #$000f,d4
303:               sub.w   d4,d3
304:               beq     hline2
305:               lsr.w   #2,d3
306:               neg.w   d3
307:               subq.w  #1,d4
308:               bcs     hline1
309: hline0:       move.w  d7,(a0)+
310:               dbra    d4,hline0
311:
312: hline1:       movem.l d5-d6/a2-a6,-(sp)
313:               move.l  d7,d5
314:               move.l  d7,d6
315:               move.l  d7,a2
316:               move.l  d7,a3
317:               move.l  d7,a4
318:               move.l  d7,a5
319:               move.l  d7,a6
320:               jmp     hlin00(pc,d3)
321:
322: hline2:       subq.w  #1,d4
323: hline3:       move.w  d7,(a0)+
324:               dbra    d4,hline3
325:               rts
326:
327:               .end
```

## リスト4 PAINT TEST.S

```
 1: *          gpaintのテスト用プログラム
 2: *
 3:            .include       doscall.mac
 4:            .include       iocscall.mac
 5: *
 6:            .xref    gpaint
 7:            .xref    setcliprect
 8: *
 9:            .offset  0
10: *
11:            .text
12:            .even
13: *
14: ent:
15:            lea.l    inisp,sp
16:
17:            move.l   #$0010_0005,-(sp)
18:            DOS      _CONCTRL
19:            addq.l   #4,sp
20:
21:            clr.l    -(sp)
22:            DOS      _SUPER
23:
24: *          pea.l    window(pc)
25: *          jsr      setcliprect
26: *          addq.l   #4,sp
27:
28:            lea.l    linarg1(pc),a1
29:            lea.l    linarg2(pc),a2
30:
31:            moveq.l  #0,d6
32:            move.w   #256-1,d7
33: loop:      move.w   d6,2(a1)
34:            move.w   d6,6(a1)
35:            IOCS     _LINE
36:            exg.l    a1,a2
37:            addq.w   #1,d6
38:
39:            move.w   d6,2(a1)
40:            move.w   d6,6(a1)
41:            IOCS     _LINE
42:            exg.l    a1,a2
43:            addq.w   #1,d6
44:            dbra     d7,loop
45:
46:            pea.l    paintarg(pc)
47:            jsr      gpaint
48:            addq.l   #4,sp
49:
50: *          lea.l    paintarg(pc),a1
51: *          IOCS     _PAINT
52:
53:            DOS      _EXIT
54: *
55:            .data
56:            .even
57: *
58: linarg1:
59:            .dc.w    0,0,511,0,$c000,$8888
60: linarg2:
61:            .dc.w    0,1,511,1,$c000,$2222
62: *
63: paintarg:
64:            .dc.w    255,255
65:            .dc.w    $003f
66:            .dc.l    paintwork
67:            .dc.l    paintworkend
68: *
69: window:    .dc.w    64,64,511-64,511-64
70: *
71:            .bss
72:            .even
73: *
74: paintwork:
75:            .ds.b    16384
76: paintworkend:
77: *
78:            .stack
79:            .even
80: *
81:            .ds.l    2048
82: inisp:
83:            .end     ent
```

# ★(で)のショートプロぱーてぃ その25

# ハリは駄菓子だ！

Komura Satoshi 古村 聡

今回は両方ともX68000用，しかもCコンパイラが必要なプログラムとあいなりました。ファイル名で遊ぶツール「EXACT.C」と，プログラムはちょっと長めだけれども，表示画面はとても小さい１行ブロック崩し「LB_ATTACK.C」です。

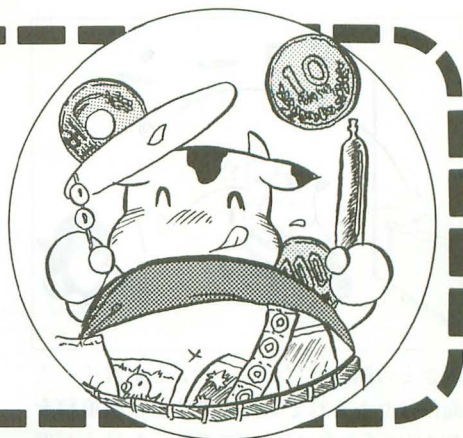



illustration : T.Takahashi

コンビニって便利ですね。いつでもあいてて，何でもあって。よく編集室でも夕食後にジュースを買いにいったり，夜中の買い出しなんかに利用するんですが，コンビニでは駄菓子も売っているんですね。

いやあ，最近の駄菓子って，“ん，ま，いっ”。昔懐かしの，のしイカをピリっと唐辛子で味つけした“甘いか太郎・キムチ味”とか，ヒゲのないドラえもんモドキの怪しげなパッケージの“うまい棒・チーズ味”とかいろいろ売っているんだけど，なかでもイチ押しなのは“ビッグカツ・とんかつソース味”ってやつ。

ああ，駄菓子の味は懐かしくておいしくて。たぶん，栄養的には何の役にも立たない，どころか，なんとなく体に悪そうな気さえするんだけど，この怪しさというか，無駄さがなんともいえない，やめられないんですよね。

それに駄菓子ってパッケージが変。パッケージの裏に書いてある解説がなんかミョーで。“蒲焼きさん太郎”っていう蒲焼き味ののしイカには“宿題のとき，１枚食べればおいしいので答えがどんどんわかります。スポーツのとき，１枚食べればおいしいので力もりもりスタミナバッチリ。これ本当の話”なんて書いてあるし。

お？ さっきのキムチ味には……，

“甘いかを現代の子供たちにあうよう，味つけしてみました。お父さんにも勧めてみてくださいね。きっと喜ばれますヨ。おこづかい増えること，これ「請け合い」”

はあ～。そんな効用まであるのか。そうだ。ねえ，編集さん，これあげるからさ……（すりすり）。

## めざせ無用の美学！

今月の１本目はX68000用，Cコンパイラを使ったプログラム，紙山さんの作品，「EXACT.C」なのです。

### EXACT.C for X68000

（要Cコンパイラ）

石川県 紙山 満

え，こいつはどんなツールだって。名前だけじゃわからない？ ふふふふふ。

使い方はものすごく簡単なんであります よ。んとですね，まずはこのプログラム，Cで書かれていますので，コンパイルしてください(囲み参照)。それからエディタで文章を作ります。なんでもいいからダダダーっと書きまくってください。それから１行が全角で９文字になるようにリターンキーで区切ってからセーブ。

で，コマンドライン上で（おっとそうだ。このプログラム，COMMAND.X上じゃないと意味がないです。VSやSX-WINDOWを使っている人はCOMMAND.Xをダブルクリックしてから実行してくださいね）

A＞EXACT 作成したファイル名

としてください。で，

A＞DIR

するってーと，あーら不思議。エディタで書いた文章をファイル名にしてしまうプログラムだったりするのです（表１）。

つまり，

A＞EXACT ファイル名

でテキストファイルの内容をファイル名として書いていきます。そうそう，書き込むディスクはカレントドライブになっていることに注意。

ふふふふふ。いいですねえ。

なぜ，このプログラムがEXACTなのか。それは，ナイアスのディスクを見るとわかります。そ，あれのマネっ子だからEXACTなのです。いいなあ，このノリ。「何の役に立つの，このプログラム？」という声もあるみたいですが，いいじゃないですか，役に立たなくて。これこそ駄菓子道に通ずる美学なんでありますよ（おいおいいつの間にそんなもんが？ しかも，強引すぎる前

表1

```
ボリュームがありません A:¥
    6 ファイル          6K Byte 使用中      1215K Byte 使用可能
 ファイル使用量         6K Byte 使用
(作ったファイル)                   79  91-08-11  14:58:06
これがエディタで書                  0  91-08-11  14:58:16
いたファイルの中身                  0  91-08-11  14:58:18
なんだよぉーん。あ                  0  91-08-11  14:58:18
ぁ，こりゃこりゃ。                  0  91-08-11  14:58:18
```

### Cプログラムの打ち込み方

まず，エディタを，

A＞ED ファイル名.C

で立ち上げてください。Cのプログラムの場合は“.”のあとは，必ず“C”です。忘れないでくださいね。おっともうひとつ。掲載されているリストを打ち込むときには，行番号を抜かして打っていきます。注意してください。で，[ESC] Eでセーブ＆終了したあと（詳しいエディタの使い方はマニュアルを参照のこと），いよいよコンパイルです。

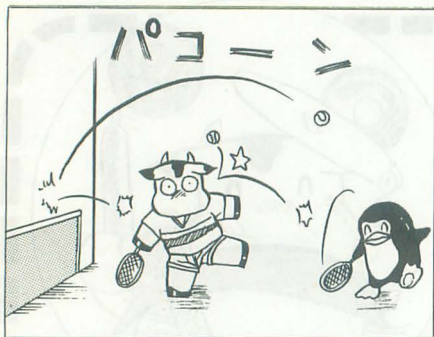
A＞CC /Y /W ファイル名.C

と打つことでコンパイルが始まります。“/Y / W”というのはBASICとDOSと同じ関数を使ってよん，ということをコンパイラに伝えるスイッチです。ショートプロの場合，ほとんどこの“/ Y /W”を使っています。最初のうちは必ずつけるようにすれば，まあ間違いないでしょう。

さて，無事，コンパイルが終われば“ファイル名.X”というプログラムができているはずです。DIRで見てみるとそのことがわかると思います。これで，

A＞ファイル名

でちゃんと実行できれば，無事完成ってわけです。

「C compiler PRO-68K」は，BASICのプログラムを“.C”の形にしてくれるBASICコンパイラも含んでいますので，BASICしかやったことがないという人でも買って損はないと思います。

置きとの関連づけ)。わ，た，しが許すっ！面白けりゃそれでいいんです。投稿原稿を見ると,「本当はBASICで作ったほうが楽だったのですが，かっこいいので(?)C言語で作ってみました」。おお，これは駄菓子のパッケージに書いてある怪しいコピーのようだ(まったく強引だなあ)。

## 1行の伝統美

さてさて，では，怪しさ大爆発したところで次のプログラムに行きましょう。今月の2本目はこれもX68000，Cコンパイラ用のプログラム(BASICユーザーの皆さん，ごめんなさい)。不思議な雰囲気の1行ブロック崩し,「LB_ATTACK.C」です。

**LB_ATTACK.C for X68000**
**(要Cコンパイラ)**
**大阪府　京野利広**

このゲーム，画面もシンプルなら，操作方法もメチャ簡単。

ゲームの最初はスタンバイ状態でボール“・”がランダムの位置に現れます。ここで，マウスの左ボタンを押すとゲームの始まり。その位置からつつつー，っとボールが左へ動き出します。そして，ボールは左端まで行くと，跳ね返ってパドル“【”のほうに向かってきます。

で，ボールがパドルの上に来たとき，すかさず「ばしっ！」とボタンを押してボールをつかまえます。すると，押している間，右端にルーレットのように次々に変化する数字が表示されます。

ボタンを放すとこの数字の変化は止まってボールが左へ跳ね返され，今度はボールは左端まで行かず，数字に対応した場所まで行って戻ってきます。つまり数字がボールの強さになってるんですね。

で，戻ってくるときに，ボールはその場所のブロックを少し崩してきます。ブロックは'□'→'○'→'○'と崩れていき，その次には消えてなくなります。もちろん，全部のブロックを消してしまうと1面クリアなのです。

ちなみに，面クリアのときや，全部のブロックを同じものに揃えるとボーナスがあります。面クリアしたり，ゾロ目にしたりするには，うまくルーレットを思った数で止めなきゃいけませんけど，ルーレットは数字がひと回りしてしまうとミスになります。欲張っちゃあいけません。

あ，マウスボタンを押すのが早すぎたり，遅すぎたりして，パドルでつかまえられなくても，当然ミスになってしまいますので注意しましょうね。

実はこのプログラム，SXエンターテイメントとして送られてきたプログラムなんです。SX-WINDOW版とコマンドライン版の両方が入ってまして,“コマンドライン版のほうは使っていいよ”ということなので使わせていただきました。投稿作品はすべて端から端まで目を通してますからね。

作者の京野さんはSX-WINDOW用にゲームをということで，なるべく小さいウィンドウでできるゲームはないだろうかと考えた。

それで思い出したのが10年ほど前にPB-100で作ったゲーム。懐かしいですね，PB-100。画面は12桁×1行，メモリは512ステップのポケットコンピュータ。画面には文字しか出ないのに，月面着陸ゲームやらゴルフゲームやらと，なんでも作っていたというポケコン時代。もうかれこれ10年近く昔の話になるんですかね。

で，10年前，中学1年生の京野少年は考えた。ブロック崩しがやりたい。やりたいったらやりたい。どうしても。で，考えついたのがこのゲームだったのだそうです。

ううむ。1行で，ものの見事にブロック崩しができてしまってるんだからたいしたもんです。さすが，若いときってちょっと古いけど“やわらか頭”ですね。

## 自由だから無駄話

ちょっと，プログラムのことなぞ。

今月のプログラムはどちらもCのプログラムだったわけですけど，両方読んでいて，Cって人のクセ(性格)が出るんだな，などと変な感心をしてしまいました。

え，お前はなんでそんな無駄なことばっかり考えるんだって？　いいもん，べつに。Oh!Xの駄菓子男と呼んでくれたまえ。その心は「役に立たない，しょーもない」。せめてOh!Xの剣先スルメぐらいにはなりたいぞ，と。まあ，編集室で唯一，X68000をペケロクと呼ぶ男でも別にいいんだけど。

話が混乱してきてしまったのでこのへんでお開き。ではまた来月。

LB_ATTACK

**リスト1　EXACT.C**

```
 1: /*=======================================
 2:
 3:                    EXACT.C
 4:
 5:   Programmed 1991 By Mitsuru Kamiyama
 6:
 7: =======================================*/
 8:
 9: #include        <stdio.h>
10: #include        <basic.h>
11: #include        <basic0.h>
12:
13: #define VER     "1.00"
14:
15: main(argc,argv)
16: int     argc;
17: char    *argv[];
18:         {
19:         int     fp1;
20:         int     fp2;
21:         char    fname[33];
22:
23:         if(argc != 2)
24:                 {
25:                 usage();
26:                 exit(0);
27:                 }
28:
29:         if((fp1 = b_fopen(argv[1],"r")) == -1)
30:                 {
31:                 printf("%s がオープン出来ません¥n",argv[1]);
32:                 exit(1);
33:                 }
34:
35:         while(b_freads(fname,sizeof(fname),fp1) != -1)
36:                 {
37:                 if( (fp2 = b_fopen(fname,"w"))  == -1)
38:                         {
39:                         printf("%s ... 書き込みエラー¥n",fname);
40:                         b_fclose(fp1);
41:                         exit(1);
42:                         }
43:                 else
44:                         {
45:                         printf("%s ... OK¥n",fname);
46:                         b_fclose(fp2);
47:                         }
48:                 }
49:         b_fclose(fp1);
50:         exit(0);
51:         }
52:
53: usage()
54:         {
55:         printf("EXACT.X for X68k version");
56:         printf(VER);
57:         printf(" By Mitsuru Kamiyama¥n");
58:         printf("使用方法:exact <filename>¥n");
59:         }
```

**リスト2** LB_ATTACK.C

```
  1: #include  <stdio.h>
  2:
  3: #define SPACE1  2         /* 0以上 */
  4: #define BLOCK  6         /* 1以上 */
  5: #define SPACE2  3         /* 1以上 */
  6: #define WLOOP   900
  7: #define WLOOPS   400
  8: #define BONUS  1000
  9: #define CBONUS  5000
 10: #define TRUE  1
 11: #define FALSE  0
 12:
 13: /*  PROTOTYPE  */
 14: int  rand(void);
 15: int  GETC(void);
 16: int  JOYGET(int);
 17: int  B_LOCATE(int,int);
 18: int  mouse(int);
 19: int  msstat(int *, int *, int *, int *);
 20:
 21: /*  INITIALIZE  */
 22: int  tx,ty;
 23: char  blchar[43][3] = {" ","○","○","□"};
 24: char  bachar[] = "・";
 25: char  pdchar[] = "【";
 26: char  w1char[] = "■",w2char[] = "■";
 27:
 28: /*追加したよ*/
 29: int  is_mouse(void)
 30: {
 31:   int  dummy,bl;
 32:
 33:   if(JOYGET(0) != 0xff) return TRUE;
 34:   msstat(&dummy,&dummy,&bl,&dummy);
 35:   if(bl == -1)  return TRUE;
 36:   else    return FALSE;
 37: }
 38:
 39: void disp(int colum,int *block)
 40: {
 41:   B_LOCATE(tx+colum*2,ty);
 42:   if (colum == 0) printf("%s",w1char);
 43:   else if (colum <= SPACE1) printf(" ");
 44:   else if (colum <= SPACE1+BLOCK) printf("%s",blchar[blo
ck[colum-SPACE1-1]]);
 45:   else if (colum <= SPACE1+BLOCK+SPACE2) printf(" ");
 46:   else if (colum == SPACE1+BLOCK+SPACE2+1) printf("%s",p
dchar);
 47:   else if (colum <= SPACE1+BLOCK+SPACE2+1+2) printf(" "
);
 48:   else if (colum == SPACE1+BLOCK+SPACE2+1+2+1) printf("%
s",w2char);
 49: }
 50:
 51: void main(void)
 52: {
 53:   int  block[BLOCK];
 54:   int  score;
 55:   int  ball;
 56:   int  pos,vec,tar;
 57:   int  trig;
 58:   int  slo;
 59:   int  def;
 60:   int  key;
 61:   int  i,j;
 62:
 63:   mouse(4);
 64:   mouse(2);
 65:
 66:   printf("¥x1b[>5h¥x0d");
 67:   for(i=1;i<=3;i++) printf("¥x1b[2K¥x0a");
 68:   printf("¥x1b[2K");
 69:   i = B_LOCATE(-1,0);
 70:   tx = 0;
 71:   ty = (i&0xffff)-3;
 72:
 73:   while(1) {
 74:     score = 0;
 75:     ball = 3;
 76:     def = 0;
 77:     slo = 0;
 78:     for(i=0;i<BLOCK;i++) block[i] = 3;
 79:     while(ball > 0) {
 80:       for(i=0;i<1+SPACE1+BLOCK+SPACE2+1+2+1;i++) disp(i,
block);
 81:       B_LOCATE(tx,ty+1);
 82:       printf("¥x1b[2K");
 83:       B_LOCATE(tx,ty+2);
 84:       printf("SCORE %10.10d",score);
 85:
 86:       trig = 0;
 87:       do {
 88:         pos = rand()%(SPACE1+BLOCK+1)+1;
 89:         B_LOCATE(tx+pos*2,ty);
 90:         printf("%s",bachar);
 91:         for(i=1;i<WLOOP-def;i++) {
 92:           if (is_mouse() == TRUE) trig = 1;
 93:         }
 94:         if (trig) break;
 95:         disp(pos,block);
 96:         for(i=1;i<WLOOP-def;i++) {
 97:           if (is_mouse() == TRUE) trig = 1;
 98:         }
 99:       } while(!trig);
100:       vec = -1;
101:       tar = 1;
102:
103:       while(1) {
104:         B_LOCATE(tx+pos*2,ty);
105:         printf("%s",bachar);
106:         if ((vec == 1) && (pos >= 1+SPACE1+BLOCK+SPACE2-
1)) {
107:           trig = 0;
108:           for(i=1;i<WLOOP-def;i++) {
109:             if (is_mouse() == TRUE) trig = 1;
110:           }
111:           if (trig) {
112:             if (pos != 1+SPACE1+BLOCK+SPACE2) {
113:               B_LOCATE(tx,ty+1);
114:               printf("MISS!");
115:               while(is_mouse() == TRUE);
116:               while(is_mouse() == FALSE);
117:               while(is_mouse() == TRUE);
118:               B_LOCATE(tx,ty+1);
119:               printf("       ");
120:               ball--;
121:               break;
122:             } else {
123:               vec = -1;
124:               B_LOCATE(tx+(1+SPACE1+BLOCK+SPACE2)*2,ty);
125:               printf("‖");
126:               slo = rand()%(SPACE1+BLOCK+1);
127:               i = 0;
128:               while(i<(SPACE2+BLOCK+1)*2) {
129:                 slo = (slo+1)%(SPACE1+BLOCK+1);
130:                 B_LOCATE(tx+(1+SPACE1+BLOCK+SPACE2+1+1)*
2,ty);
131:                 printf("¥x82%c",0x4f+slo);
132:                 j = 1;
133:                 while(j<WLOOPS-def) {
134:                   if (is_mouse() == FALSE) {
135:                     trig = 0;
136:                     i = (SPACE2+BLOCK+1)*2;
137:                     break;
138:                   }
139:                   j++;
140:                 }
141:                 i++;
142:               }
143:               if (trig) {
144:                 B_LOCATE(tx+(1+SPACE1+BLOCK+SPACE2+1+1)*
2,ty);
145:                 printf("×");
146:                 B_LOCATE(tx,ty+1);
147:                 printf("MISS!");
148:                 while(is_mouse() == TRUE);
149:                 while(is_mouse() == FALSE);
150:                 while(is_mouse() == TRUE);
151:                 B_LOCATE(tx,ty+1);
152:                 printf("       ");
153:                 ball--;
154:                 break;
155:               }
156:               tar = 1+SPACE1+BLOCK+1-slo-1;
157:               continue;
158:             }
159:           }
160:           if (pos >= 1+SPACE1+BLOCK+SPACE2+1) {
161:             B_LOCATE(tx,ty+1);
162:             printf("MISS!");
163:             while(is_mouse() == TRUE);
164:             while(is_mouse() == FALSE);
165:             while(is_mouse() == TRUE);
166:             B_LOCATE(tx,ty+1);
167:             printf("       ");
168:             ball--;
169:             break;
170:           }
171:         } else {
172:           trig = 0;
173:           for(i=1;i<WLOOP-def;i++) {
174:             if (is_mouse() == TRUE) trig = 1;
175:           }
176:           if ((vec == -1) &&(pos == tar)) {
177:             if ((pos >= 1+SPACE1) && (pos <= 1+SPACE1+BL
OCK-1)) {
178:               if (block[pos-(1+SPACE1)] > 0) {
179:                 B_LOCATE(tx,ty+1);
180:                 printf("       ");
181:                 score += (4-(block[pos-(1+SPACE1)]--))*1
0;
182:                 B_LOCATE(tx,ty+2);
183:                 printf("SCORE %10.10d",score);
184:                 for(i=j=0;i<BLOCK;i++) {
185:                   if (block[0] == block[i]) j++;
186:                 }
187:                 if (j == BLOCK) {
188:                   if (block[0] == 0) {
189:                     B_LOCATE(tx,ty+1);
190:                     printf("CLEAR! BONUS!!");
191:                     score += CBONUS;
192:                     B_LOCATE(tx,ty+2);
193:                     printf("SCORE %10.10d",score);
194:                     while(is_mouse() == TRUE);
195:                     while(is_mouse() == FALSE);
196:                     while(is_mouse() == TRUE);
197:                     for(i=0;i<BLOCK;i++) block[i] = 3;
198:                     def += 50;
199:                     break;
200:                   } else {
201:                     B_LOCATE(tx,ty+1);
202:                     printf("BONUS!");
203:                     score += (4-block[0])*BONUS;
204:                     B_LOCATE(tx,ty+2);
205:                     printf("SCORE %10.10d",score);
206:                   }
207:                 }
208:               }
209:             }
210:             vec = 1;
211:             continue;
212:           }
213:         }
214:         disp(pos,block);
215:         pos += vec;
216:       }
217:     }
218:     B_LOCATE(tx,ty+1);
219:     printf("REPLAY? (Y/N)");
220:     do {
221:       key = GETC()|0x20;
222:     } while((key != 'y') && (key != 'n'));
223:     if (key == 'n') break;
224:   }
225:
226:   B_LOCATE(0,ty+2);
227:   printf("¥x0a¥x1b[2K¥x1b[>51");
228:   mouse(0);
229: }
```

# (で)のぱーてぃハンズ第3部――(その4)

はい，では先月の考え方で，うまくコンピュータにあわせて思考方法を考えていきます。

まず，手を選ぶときにいちばん勝てそうな手を選べばよいという話はしましたよね。で，いちばん勝てそうな手を選ぶには，そのあとの手を見ていく，と。

で，コンピュータにどれがいちばん勝てそうか，というのを判断させるわけですが，これを判断させるには，なんとなく勝てそうとか，ちょっとまずいな，なんてのはまずいわけです。

それではどうしたものか。昔の偉い人は考えた。そうだ。どのくらい勝てそうか，というのを数字で表せばいいのだ。つまり，0番の手を選んだときには0点，1番なら100点，2番なら50点……，というふうに表すことができれば，1番がいちばん勝てそうだなとひと目でわかるわけです。次に打つべき手を選ぶなんてのはもうお茶の子さいさいなのですね。

で，昔の偉いおっさんはこの，「どのくらい勝てそうだな」という点数のことを評価値と名づけたのです。

ルール1：どのくらい勝ちに結びつきそうな手であるかを数字で表す

## こわい考えになる前に……

ところで，ゲームオーバーになるまで打てる手をかたっぱしからコンピュータに調べさせると話しました。しかし，たしかに人間が計算するよりはずいぶん計算が早いけど，ゲームの場合は手の数がとんでもなく多すぎるのです。

○×ゲーム（ティク・タク・トゥ）だと，ゲームオーバーになるまで何手になるか考えてみると，先手後手あわせても9×8×7×6×5×4×3×2×1＝362880通りしかないから，パソコンでも全部手が読めそうです。

しかし，これが将棋の場合だったりすると，えっと，まず，王が8通り動けるでしょ。飛車が16……。とかいって，考えていくだけでも気が遠くなりそうなほどの手が存在します。真面目にやったら何万年とかかるんじゃないかと思ってしまいそうです。

つまり，この方法はすべてのゲームに対しては有効でない，というか，普通の終わりまで何手も必要とするようなゲームではまず，ゲームができるほど早いスピードでは手を打つことができない，ということになってしまいます。

それではどうしようか？ なにか時間を短縮する方法はないか……，と考えるとゲームオーバーまで手を読み切っていたのを途中でプチッと切って妥協してしまうのが簡単そうです。正確な評価値は求められませんけどね。

ルール2：思考の深さは適当なところで切ってしまう

## 評価値を求める

では，ゲームの最後まで考えないでどうやって評価値を求めるんでしょうか？

実はここで“評価関数”というものを作ってやるのです。これは1手ごとにその手が打つ前の状態に比べてどのくらい有利になるかということを数値にして表す関数です。しかし，この関数の作り方がミソになります。たとえばですね，将棋やチェスをやっている途中で局面を見せられて，どのくらい勝てそうか数字でいってねといわれて，はっきりいくつだよと答えられますか。どの駒にどのくらい重きをおいているかによって違いますしね。

で，対処法としては……，皆さん，適当に作ってるみたいです。というか，これぱっかりは試行錯誤でいろいろやってみるしかないんですよね。最初はまあ考えつくかぎり簡単な評価関数を作ってみて，徐々に複雑なものにしていくというのが普通みたいなんですが。

思考ゲームの場合はこの評価関数の出来がすべてといってもいいくらいで，しかも人それぞれいろいろなものができてくるので，まさしくこれが思考ルーチンの醍醐味といえましょう。

ルール3：評価関数が手を打つ前と打った後の差を評価値として出す

## 評価値はどんどん変わる

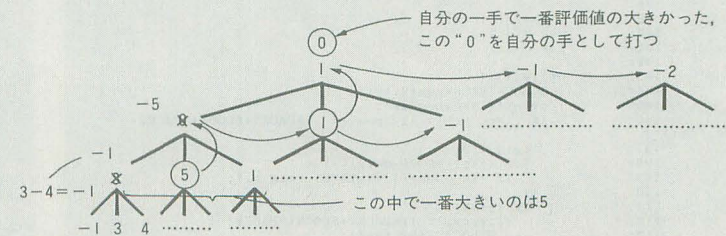
さらに話はややこしくなります。

問題は敵の存在です。評価値を求めるときには，その手を打つと敵が次に手を打ってくる，それがどのくらい敵に有利になっているか，つまり敵にとっての評価値がいくらになっているかを考えなくてはいけないのです。

```
     5            3         自分の打つ手
   ／|＼        ／|＼
 1  −3  4     0  1  −1      敵の打つ手
```

（評価値は当然評価関数が出した，その打った本人にとって，その手を打つとどのくらい勝利に近づくかを数字で表した数値です）

上のような場合，自分の打つ手はどちらが有利になるんでしょう。自分の打つ手だけを考えれば，5点のほうが有利な気がしてきますね。ところが，敵が次にどんな手を打てるかを考えると……，まず，左側。敵は1，−3，4点のうちどれを取るか。いちばんいい手は4点の手ですから，当然それを取ってくるでしょうね。つまり，自分，敵がそれぞれ手を打った結果は5点勝利に近づいて，4点勝利から遠ざかったから（敵が勝利に近づく＝自分の勝利が遠のくでしょ），結局，5−4＝1点分しか勝利に近づいていないのですね。ところが右側を見てください。自分は3点しか取れませんが，敵のとれるいちばんいい手も1点でしかないんです。つまり3−1＝2点分勝利に近づいているので，右側のほうが自分にとって有利な手なのです。

ルール4：自分と敵とは交互に手を打っていくので，自分の評価値から敵の評価値を引いたものが，より正しい評価値である

## そしてミニマックスへ

では，これまで登場した4つのルールを使って，思考ルーチンを追いかけてみます。“適当な深さで手の探索をやめる”のですが，ここではいちおう4手までとします（図）。

1） いちばん上の左側に，1点という，とりあえずの評価値が出ています

2） で，その次に敵の取れる3つの手について考えなければいけません。まず，1番目の手について考えてみると，評価値は0と出ました

3） ところが，敵にとってもこれは正しい評価値ではありません。敵の場合も自分の評価値を引かないと正しい評価値が出てこないからです

4） 次の自分の3つの評価値が必要ですが，それも正しくないので，さらに下を見にいきます

5） ここでも本当はその次を求めなくては正しい評価値が出ないのですが，4手までで打ち切ることにしていたので，しかたなくその時点での評価値，つまり，−1，3，4で妥協します

6） そのなかで最大の評価値は4です。つまり，この状況になったら敵は4点得をする手を打ってくるに違いない，と考えるわけです

7） さて，では−1，3，4の上の自分の3点に戻ってきます。この手を取ると3点自分が取れて，敵は4点，そのあとに取る。この手を打ったときの本当の評価値は3−4＝−1点となります

8） こういうふうにして，1手目まで戻っていくと，この手においては0という評価値を出すことができました

9） 同様にほかの手の評価値も求めてみると−1，−2点となり，結局最初に求めた手の評価値，0点が最高ということになりました

これこそが我々の目指していたミニマックス法による思考ルーチンの動きなのです。

## さあ，組むぞ!

というわけでこういう動きをするプログラムを組めばいいだけですね。

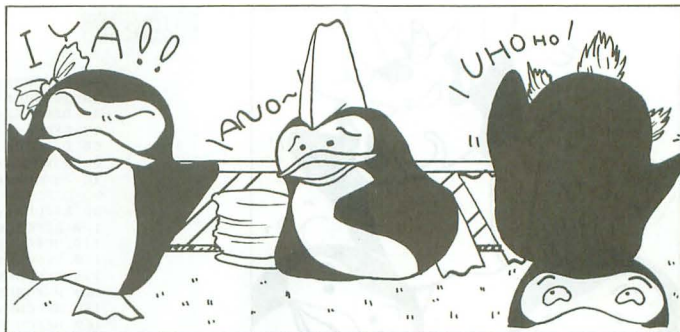
さて，それではさっき書いた木をずっと下まで降りていって，また上がってくるようなプログラムを組むにはどうしたらいいんだ。そう，あれですよ，あれ。ふふふふふ。というわけで，いよいよ来月は第3部の最終回。リストつきでミニマックスプログラムの解説，さらに思考時間を短縮するための枝刈りの話なんかもしたいと思っています。ああ，今月もリストなし。

しかし，いよいよ来月で終わり。さあ，しまっていくぞ。おう。まさか，ゲームの内容を忘れちゃいないだろうなあ。また来月。

図

☆点数は自分，敵それぞれ打った者が得る点数

# マシン語カクテル in Z80's Bar
## 第25回─秋の運動会スペシャル─

投稿プログラム：遠藤亮司
シナリオ　　　：金子俊一
特別監修　　　：浦川博之

今月はいつもとちょっと違います。なんと，「Z80's Bar」のゲームが投稿されてきたのです。ということで，今回は秋の運動会スペシャルと銘打って，そのプログラムをどどーんと載せてしまいます。X1ユーザーの方はぜひぜひ打ち込んで遊んでみてくださいね。

♪カラン，コロ～ン
**源光（以下光）**：こんにちは。
**ようこ（以下Yo）**：あら，原稿は持ってきた？
**光**：なにをいきなり？
**Yo**：だから原稿よ，げ・ん・こ・う。ちゃんと12枚集めたの？
**マスター（以下M）**：ようこちゃんもハマりましたね。
**光**：何にハマったんですか？
**長老（以下老）**：ゲームじゃよ。投稿が届いたんじゃ。
**光**：ほう。
**老**：なかなかしっかりしたゲームじゃよ。マシン語で長いプログラムを作ったのは初めてらしいのじゃが，そうとは思えんのう。ストーリーも面白いしのう。
**光**：どんなストーリーなんですか？
**Yo**：知らないほうが幸せなんじゃない？
**光**：教えてくださいよ。
**老**：それでは，ストーリーと遊び方をまとめて紹介しようかの。

## ストーリー

ある日，「Z80's Bar」で光君は長老やほかの常連客と賭けをしました。それは「光君がようこちゃんをデートに誘って，OKをもらえるかどうか」というものでした。
**光**：誘えるほうに500カノッサ。
もし，誘うことができなければ，みんなのツケを払うという約束をしていました。
**光**：ねえ，ようこちゃん，今度の日曜日デートにいかない？
**Yo**：いや！
**光**：がび～～～ん。
偶然にも賭けの話を聴いていたようこちゃんがOKを出すはずがありません。
ついてない光君は締め切り間際にもかかわらず，プログラムを作って原稿を書いてこなければなりません。3日間の徹夜の末，やっと出来上がった原稿。しかし，「Z80's Bar」に持っていく途中で，またもやついてない光君は原稿を風に飛ばされてしまいました。
締め切りの時間が過ぎてもやってこない光君に，しびれを切らした常連客たちは光君を探しにいくことになりました。
光君は常連客に見つからないように原稿を集めて，「Z80's Bar」に持っていき，ツケを払わなければなりません。

## 遊び方

12枚の原稿を集めて「Z80's Bar」に戻ると1面がクリアになり，面数は全部で10面あります。12枚揃わずに「Z80's Bar」に戻ると，ゲームオーバーになります。
コインを拾って自動販売機に入れると，牛に変身することができ，常連客たちを病院送りにすることができます。音が変わっている間だけ強くなります。動かないと変身したり，戻ったりできません。
ボーナスステージが2面おきにあります。好きなキャラクタの真下に来てから，トリガを押してください。キャラクタの下にボーナスが隠れています。
ちなみに，ジョイスティック専用です。

＊　＊　＊

**光**：ちょっと待ってくださいよ。
**老**：どうしたのじゃ。
**光**：遊び方はともかく，このストーリーはなんですか。
**Yo**：あら，ありそうな話じゃない。500カノッサってあたりがTVブックメーカーの見すぎって感じでいいわ。
**光**：何いってるんですか，ようこさんまで。
**Yo**：面白そうでしょ？
**光**：そりゃ，まあ。
**Yo**：それじゃあ入力するしかないわ。私が入力方法を説明してあげるから。

## 入力方法

**Yo**：X1シリーズとCZ-8FB01を用意してください。
リスト1はPCGのセットプログラム，リスト2がマシン語のローダ，リスト3がプログラム本体になっています。

**リスト1**

```
10 WIDTH 40:CLS4:INIT:COLOR7:CGEN1
20 DEFCHR$(35)=HEXCHR$("5AFF7EFFFF7EFF5A5A8100818100815A5A8100818100815A")
30 LINE (0,0)-(39,24),"#",BF:INIT
40 LINE (2,1)-(29,22)," ",BF
50 COLOR3:LOCATE31,2:PRINT"Z80'sBar"
60 COLOR6:LOCATE31,5:PRINT"HIGH":LOCATE32,6:PRINT"SCORE"
70 LOCATE31,10:PRINT"SCORE"
80 LOCATE31,14:PRINT"ROUND":Y=8
90 FOR T=1 TO 2
100 LOCATE 32,Y:PRINT"      ":Y=Y+4:NEXT
110 LOCATE 35,19:PRINT"  "
120 LOCATE 35,22:PRINT"  "
130 RUN"PCG.Bas"
```

リスト3の入力は慎重に行いましょう。入力し終わったら，まずセーブすることが大切です。

リスト2の“PCG.Bas”とリスト3の“マシンゴ.bin”ファイルネームを間違えないようにしましょう。

リスト1をRUNすると，プログラムの3段ロードでデモが始まります。

**光**：ふむふむ。

**Yo**：どう，面白い？

**光**：ちょっとジョイスティックの反応が鈍いのが気にかかりますね。

**老**：それがまたゲームを面白くしとるじゃろう。

**光**：この「じゅんじ」ってヤローが妙にしつこいですね。

**Yo**：「長老」や「ぜんちゃん」もいい味だしてるわよ。

**老**：ふぉっふぉっふぉ。「光君」も似とるしのう。

**Yo**：メアリーがいるわりには，私がいないわ。

**M**：お店はどうするの，お店は。

**Yo**：あっ，そうか。

**光**：ようこさんにも困ったもんですね。

―つづく―

### リスト2

```
10 DEFCHR$(65)=HEXCHR$("00000103030F1F7F00000103030100 6F00000103030100 6F")
20 DEFCHR$(66)=HEXCHR$("0000008080E0F0FC0000008080000 0EC00000080800000EC")
30 DEFCHR$(67)=HEXCHR$("FF7D301C7F230100DF4D0F0307030100DF4D0F03071F3F3E")
40 DEFCHR$(68)=HEXCHR$("FE7C1070FCF0F000F674EE84C0C4C404F674E080C0C0C0F8")
50 DEFCHR$(69)=HEXCHR$("00000103030F1F7F000001030301006F000001030301006F")
60 DEFCHR$(70)=HEXCHR$("0000008080E0F0FC00000080800000EC00000080800000EC")
70 DEFCHR$(71)=HEXCHR$("FF7D301C1F3F7F00DF4D0F03070707 00DF4D0F030707073E")
80 DEFCHR$(72)=HEXCHR$("FE740850F8800000F67CF0A8C4820100F674E080C0F0F8F8")
90 DEFCHR$(73)=HEXCHR$("000000000070F1F1F00000000070F1F1E00000000000000030F")
100 DEFCHR$(74)=HEXCHR$("00002060F8F8FCFC00000000E0E8FCBC000020601810E0F8")
110 DEFCHR$(75)=HEXCHR$("1F181E7F3F1F1F001F1F1F7F3F1F1F000F070103070 7033E")
120 DEFCHR$(76)=HEXCHR$("FC0C3CFCFEE3C000FCFCFCFCFEE3C000F8F0C0F0F0FCFE3E")
130 DEFCHR$(77)=HEXCHR$("000000000070F1F1F00000000070F1F1E00000000000000030F")
140 DEFCHR$(78)=HEXCHR$("00002060F8F8FCFC00000000E0E8FCBC000020601810E0F8")
150 DEFCHR$(79)=HEXCHR$("1F181E1F3F6301001F1F1F1F3F6301000F070107071F3F3E")
160 DEFCHR$(80)=HEXCHR$("FC0C3CFFFEFCFC00FCFCFCFFFEFCFC00F8F0C0E0F0F0F03E")
170 DEFCHR$(81)=HEXCHR$("000000000070F1F1F000000000000060E000000000000000060E")
180 DEFCHR$(82)=HEXCHR$("00000000E0F0F8F800000000000060700000000000000006070")
190 DEFCHR$(83)=HEXCHR$("1D181EFF7F3F1F000E07010307070300 0C070103070703 7C")
200 DEFCHR$(84)=HEXCHR$("B81878FCFEE0C00070E080C0E0E0C00030E080C0E0FCFE3E")
210 DEFCHR$(85)=HEXCHR$("000000000070F1F1F000000000000060E000000000000000060E")
220 DEFCHR$(86)=HEXCHR$("00000000E0F0F8F800000000000060700000000000000006070")
230 DEFCHR$(87)=HEXCHR$("1D181E3F7FC703000E0701030707030000C070103073F7F7C")
240 DEFCHR$(88)=HEXCHR$("B81878FFFEFCF80070E080C0E0E0C00030E080C0E0E0C03E")
250 DEFCHR$(89)=HEXCHR$("000000000A1F3F3D000005050500071F000005050500071D")
260 DEFCHR$(90)=HEXCHR$("00000000A0F0F87800000404040 00C0F000004040400 0C070")
270 DEFCHR$(91)=HEXCHR$("3F303CFF7F3F3F001F0F030F0F0F0700130F030F0F0F077C")
280 DEFCHR$(92)=HEXCHR$("F81878F8F0C28000F0E081C6EECC800090E080C0E0F0FC7C")
290 DEFCHR$(93)=HEXCHR$("000000000A1F3F3D000005050500071F000005050500071D")
300 DEFCHR$(94)=HEXCHR$("00000000A0F0F878000004040400 0C0F000004040400 0C070")
310 DEFCHR$(95)=HEXCHR$("3F303C3F7FC703001F0F030F0F070300130F030F0F3F7F7C")
320 DEFCHR$(96)=HEXCHR$("F81878F8F0E2F000F0E081C6EEFCE80090E080C0E0E0E07C")
330 DEFCHR$(97)=HEXCHR$("000000000000030F0001010107 0F1F1E000101010100030F")
340 DEFCHR$(98)=HEXCHR$("0000000000 00E0F880C0C0C0F0F8FCBC80C0C0C0C080E0F8")
350 DEFCHR$(99)=HEXCHR$("0F000007C783C1001F1F1F1F3F6301000F070107C79FFF3E")
360 DEFCHR$(100)=HEXCHR$("F80000E0F0F0F000FCFCFCFFFEFCFC00F8F0C0E0F0F0F03E")
370 DEFCHR$(101)=HEXCHR$("000000000000030F01030303070F1F1E01030303 0100030F")
380 DEFCHR$(102)=HEXCHR$("000000000000E0F8008080C0F0F8FCBC008080C0C080E0F8")
390 DEFCHR$(103)=HEXCHR$("0F00C087C70703001F1F1F7F3F1F1F000F07C187C707033E")
400 DEFCHR$(104)=HEXCHR$("F80000E0F0E0C000FCFCFCFCFEE3C000F8F0C0E0F0FCFE3E")
410 DEFCHR$(105)=HEXCHR$("02050A152A552A550000000000000000030F1F3F7F7F7F7F")
420 DEFCHR$(106)=HEXCHR$("8050A854AA54AA540000000000000000C0F0F8FCFEFEFEFE")
430 DEFCHR$(107)=HEXCHR$("22101004000000000090D070303030 3077F3F1F0F03030307")
440 DEFCHR$(108)=HEXCHR$("2804100000000000090B0E0C0C0C0C0E0FEFCF8F0C0C0C0E0")
450 DEFCHR$(109)=HEXCHR$("FF80808080808080000003C051819213CFFFFC3FAE7E6DEC3")
460 DEFCHR$(110)=HEXCHR$("FF010101010101010000CC32D23232CCFFFF33CD2DCDCD33")
470 DEFCHR$(111)=HEXCHR$("C07F7F7F7F7F7F00003F3F26263F3F00FF7F7F7F7F7F7FFF")
480 DEFCHR$(112)=HEXCHR$("03FEFEFEFEFEFE0000FCC4444CC4C400FFFEFEFEF6FEFEFF")
490 DEFCHR$(113)=HEXCHR$("003F3F313F3F3F3F003F3F3E3F323F31003F3F303F323F31")
500 DEFCHR$(114)=HEXCHR$("00FCFC4CFCFCFCFC00FCFCBCFCA4FC4C00FCFC0CFCA4FC4C")
510 DEFCHR$(115)=HEXCHR$("3F3F3F3F3F3F3F003F343F323F393F003F343F323F393F00")
520 DEFCHR$(116)=HEXCHR$("FCFCFCFCFCFCFC00FC54FC94FC44FC00FC54FC94FC44FC00")
530 DEFCHR$(117)=HEXCHR$("7F40404F4F4E4E40003F3F3F3F3E3E3F0000000F0F0A0A00")
540 DEFCHR$(118)=HEXCHR$("FE0202F2F2B2B20200FCFCFCFCBCBCFC000000F0F0F0F000")
550 DEFCHR$(119)=HEXCHR$("404040404747404 03A3F3F3F38383F3F050000000000000000")
560 DEFCHR$(120)=HEXCHR$("12120202E2E20202ACECFCFC1C1CFCFC50100000000000000")
570 DEFCHR$(121)=HEXCHR$("000000070E0C0C0E00000070F0F0F0F000000070E0C0C0E")
580 DEFCHR$(122)=HEXCHR$("000000E070303070000000E0F0F0F0F0000000E070303070")
590 DEFCHR$(123)=HEXCHR$("FFFFFFFFFFFFFFFFFFFF9292FF9292FEFFFFFFFFFFFFFEFEFE")
600 DEFCHR$(124)=HEXCHR$("FFFFFFFFFFFFFFFFFFFF4949FF39393FFFFFFFFFFFF3F3F3F")
610 DEFCHR$(125)=HEXCHR$("030C1020408080800030F1F3F7F7F7F00030F1F3F7E717F")
620 DEFCHR$(126)=HEXCHR$("C030080402010101 00C0F0F8FCFEFEFE00C0F0F8FC1EBE56")
630 DEFCHR$(127)=HEXCHR$("8080804020100C037F7F7F3F1F0F03007E7E7E3F1F0F0300")
640 DEFCHR$(128)=HEXCHR$("0101010204083 0C0FEFEFEFCF8F0C000FEFEFE7CB8F0C000")
650 DEFCHR$(129)=HEXCHR$("000000F77F1F0F0B000206C46C1E0F0F000206C46C1E0F0B")
660 DEFCHR$(130)=HEXCHR$("000000DEFCF0E0A00080C0C6ECF0E0E00080C0C6ECF0E0A0")
670 DEFCHR$(131)=HEXCHR$("3D7F7F7F7FFFFFFF80F1F3F1D7BF8F8F82D5F3F5D0B040700")
680 DEFCHR$(132)=HEXCHR$("78FCFCFCFCFEFE1EE0F0F870A000001E68F4F874B060E000")
690 DEFCHR$(133)=HEXCHR$("000000F77F1F0F0B000206C46C1E0F0F000206C46C1E0F0B")
700 DEFCHR$(134)=HEXCHR$("000000DEFCF0E0A00080C0C6ECF0E0E00080C0C6ECF0E0A0")
710 DEFCHR$(135)=HEXCHR$("3D7F7F7F7FFFFFFF00F1F3F1D0B0000F02D5F3F5D0B0C0F00")
720 DEFCHR$(136)=HEXCHR$("78FCFCFCFCFEFE3EE0F0F870BC3E3E3E68F4F874A040C000")
730 LOADM"ﾏｼﾝｺﾞ.bin"
740 CALL &HE000
```

### リスト3

```
E000 DD 21 93 F0 3E 00 DD 77 :13
E008 00 21 A1 F0 77 3E 41 DD :85
E010 21 71 F0 CD 18 E3 DD 21 :48
E018 64 F0 CD 18 E3 DD 21 57 :71
E020 F0 CD 18 E3 DD 21 4A F0 :F0
E028 CD 18 E3 DD 21 7E F0 CD :01
E030 18 E3 DD 21 93 F0 3E 81 :3B
E038 CD 18 E3 3E 01 32 A4 F0 :CD
E040 3E 00 32 91 F0 32 92 F0 :A5
E048 26 00 2E 00 22 9F F0 3E :43
E050 03 32 AB F0 CD DB E2 3E :98
E058 00 32 A3 F0 CD C9 E8 CD :10
E060 6F E1 CD A9 E2 CD C6 E6 :21
E068 3A A4 F0 32 A3 F0 CD C9 :29
E070 E8 DD 21 71 F0 3E 02 DD :64
E078 77 00 3E 1C 67 DD 77 01 :8D
-----------------------------------
SUM: 73 49 76 BD CA 0C 90 C0 :15

E080 3E 15 6F DD 77 02 3E 0A :60
E088 DD 77 0C CD 28 E8 DD 21 :3B
E090 64 F0 3E 02 DD 77 00 3E :26
E098 02 67 DD 77 01 3E 15 6F :80
E0A0 DD 77 02 3E 1E DD 77 0C :12
E0A8 CD 28 E8 DD 21 57 F0 3E :60
E0B0 02 DD 77 00 3E 02 67 DD :DA
E0B8 77 01 3E 01 6F DD 77 02 :7C
E0C0 3E 28 DD 77 0C CD 28 E8 :A3
E0C8 DD 21 4A F0 3E 01 6F DD :C3
E0D0 77 00 DD 77 02 3E 1C DD :04
E0D8 77 01 67 3E 14 DD 77 0C :91
E0E0 CD 28 E8 DD 21 7E F0 26 :6F
E0E8 0E 2E 09 CD 28 E8 3A A3 :FF
E0F0 F0 CD BC E2 CD 62 E1 CD :38
E0F8 A9 E2 CD 39 E3 DD 21 4A :BC
-----------------------------------
SUM: 21 AF 1A 20 C2 40 CB 8F :66

E100 F0 CD A6 E3 DD 21 57 F0 :8B
E108 CD C2 E3 CD A9 E4 DD 21 :CA
E110 64 F0 CD 90 E3 DD 21 71 :03
E118 F0 CD 90 E3 CD 62 E1 3A :7A
E120 AD F0 E6 FF CA 34 E1 3D :9E
E128 32 AD F0 21 8E EF CD A7 :E1
E130 ED C3 FA E0 DD 21 93 F0 :0B
E138 DD 7E 00 FE 0F C2 59 E1 :64
E140 DD 66 01 DD 6E 02 DD 21 :8F
E148 7E F0 DD 74 01 DD 75 02 :14
E150 DD 21 93 F0 3E FF DD 77 :12
E158 00 21 1A F0 CD A7 ED C3 :4F
E160 FA E0 3A A5 F0 4F 06 FF :FD
E168 10 FE 0D C2 66 E1 C9 26 :13
E170 0D 2E 03 CD A5 E8 CD C1 :26
E178 E8 06 0A 11 B1 F0 CD 72 :E9
-----------------------------------
SUM: F1 D4 95 97 A0 D7 55 26 :E3

E180 E2 26 0C 2E 04 CD A5 E8 :A0
E188 CD C1 E8 06 0C 11 BB F0 :44
E190 CD 72 E2 26 0A 2E 07 CD :53
E198 A5 E8 CD C1 E8 06 0C 11 :26
E1A0 C7 F0 CD 72 E2 26 0A 2E :36
E1A8 0A CD A5 E8 CD C1 E8 06 :E0
E1B0 05 11 D3 F0 CD 72 E2 26 :20
E1B8 0A 2E 0D CD A5 E8 CD C1 :2D
E1C0 E8 06 07 11 D8 F0 CD 72 :0D
E1C8 E2 26 0A 2E 10 CD A5 E8 :AA
E1D0 CD C1 E8 06 05 11 DF F0 :61
E1D8 CD 72 E2 26 0A 2E 13 CD :5F
E1E0 A5 E8 CD C1 E8 06 05 11 :1F
E1E8 E4 F0 CD 72 E2 DD 21 93 :86
E1F0 F0 3E 1A DD 77 01 3E 03 :DE
E1F8 DD 77 02 06 12 C5 CD 79 :79
-----------------------------------
SUM: BB 29 86 B3 6D F8 A9 08 :33

E200 E2 CD 62 E1 C1 10 F6 DD :96
E208 21 7E F0 3E 17 DD 77 01 :39
E210 3E 06 DD 77 02 CD 5F E2 :A8
E218 DD 21 57 F0 3E 17 DD 77 :EE
E220 01 3E 06 3E 09 DD 77 02 :E2
E228 CD 5F E2 DD 21 71 F0 3E :AB
E230 17 DD 77 01 3E 0C DD 77 :0A
E238 02 CD 5F E2 DD 21 64 F0 :62
E240 3E 17 DD 77 01 3E 0F DD :D4
```

```
E248 77 02 CD 5F E2 DD 21 4A :CF
E250 F0 3E 17 DD 77 01 3E 12 :EA
E258 DD 77 02 CD 5F E2 C9 06 :33
E260 11 C5 CD 79 E2 CD 62 E1 :0E
E268 21 3C EF CD A7 ED C1 10 :7E
E270 F0 C9 1A 77 13 23 10 FA :8A
E278 C9 DD 66 01 DD 6E 02 25 :7F
---------------------------------
SUM: 72 2E 43 C2 8F 95 BD 2D :B3

E280 CD 28 E8 DD 66 01 DD 6E :6C
E288 02 24 24 CD A5 E8 CD C1 :32
E290 E8 7E CD C1 E9 DD 66 01 :21
E298 DD 6E 02 24 24 2C CD A5 :33
E2A0 E8 CD C1 E8 7E CD C1 E9 :53
E2A8 C9 CD 62 E1 01 00 1C 3E :34
E2B0 0E ED 79 05 ED 78 E6 20 :E4
E2B8 C8 C3 A9 E2 01 0A 00 26 :47
E2C0 00 6F CD 36 E7 E5 F5 26 :59
E2C8 24 2E 10 CD A5 E8 F1 E1 :8E
E2D0 CD C1 E9 03 7D C6 30 CD :BA
E2D8 C1 E9 C9 26 1F 2E 12 CD :C5
E2E0 A5 E8 DD 21 7E F0 CD 4B :11
E2E8 E8 26 24 2E 13 CD A5 E8 :CD
E2F0 3A AB F0 C6 30 CD C1 E9 :42
E2F8 DD 21 92 F0 26 1F 2E 15 :08
---------------------------------
SUM: 71 A3 32 70 94 AB 29 14 :32

E300 CD A5 E8 3E 7D CD 7C E9 :47
E308 DD 7E 00 C6 30 26 24 2E :C9
E310 16 CD A5 E8 CD C1 E9 C9 :B0
E318 DD 77 04 3C DD 77 05 3C :29
E320 DD 77 06 3C DD 77 07 3C :2D
E328 DD 77 08 3C DD 77 09 3C :31
E330 DD 77 0A 3C DD 77 0B 3C :35
E338 C9 DD 21 93 F0 DD 7E 00 :A5
E340 FE 0F CA 49 E3 DD 21 7E :7F
E348 F0 01 00 1C 3E 0E ED 79 :BF
E350 05 ED 78 32 8C F0 E6 02 :00
E358 CA 74 E3 3A 8C F0 E6 01 :BE
E360 CA 7B E3 3A 8C F0 E6 04 :C8
E368 CA 82 E3 3A 8C F0 E6 08 :D3
E370 CA 89 E3 C9 CD D5 E7 CD :55
E378 73 E4 C9 CD AB E7 CD 73 :BF
---------------------------------
SUM: 8B 84 61 4A A7 D4 81 16 :CC

E380 E4 C9 CD 56 E7 CD 73 E4 :DB
E388 C9 CD 80 E7 CD 73 E4 C9 :EA
E390 CD 44 E7 C2 A9 E4 CD 8F :A3
E398 E8 DD 46 00 B8 CA 96 E3 :06
E3A0 DD 77 00 C3 90 E3 3A AF :73
E3A8 F0 FE 0E CA B9 E3 CD 90 :BF
E3B0 E3 3A AF F0 3C 32 AF F0 :C9
E3B8 C9 CD 96 E3 3E 00 32 AF :2E
E3C0 F0 C9 3A B0 F0 FE 00 C2 :53
E3C8 68 E4 FD 21 7E F0 DD 7E :33
E3D0 02 FD 46 02 B8 D2 DE E3 :92
E3D8 DD 46 02 FD 7E 02 98 57 :91
E3E0 DD 7E 01 FD 46 01 B8 D2 :2A
E3E8 F0 E3 DD 46 01 FD 7E 01 :73
E3F0 98 BA DA 2A E4 DD 7E 01 :96
E3F8 FD 46 01 B8 DA 09 E4 CA :8D
---------------------------------
SUM: 74 84 05 54 81 8C 8D 15 :00

E400 5F E4 CD 56 E7 C0 C3 0D :DD
E408 E4 CD 80 E7 C0 DD 7E 02 :35
E410 FD 46 02 B8 CA 5F E4 D2 :DC
E418 22 E4 CD D5 E7 C0 CD 5F :7B
E420 E4 C9 CD AB E7 C0 CD 5F :F8
E428 E4 C9 DD 7E 02 FD 46 02 :4F
E430 B8 CA 5F E4 D2 3E E4 CD :86
E438 D5 E7 C0 C3 42 E4 CD AB :DD
E440 E7 C0 DD 7E 01 FD 46 01 :47
E448 B8 DA 57 E4 CA 5F E4 CD :A7
E450 56 E7 C0 CD 5F E4 C9 CD :A3
E458 80 E7 C0 CD 5F E4 C9 3E :3E
E460 06 32 B0 F0 CD 90 E3 C9 :E1
E468 3A B0 F0 3D 32 B0 F0 CD :B6
E470 90 E3 C9 FD 21 93 F0 FD :DA
E478 7E 00 FE 0F CA 8C E4 21 :E6
---------------------------------
SUM: 7A 4B 00 CF C8 1E 19 A6 :39

E480 56 EF CD A7 ED FD 21 7E :42
E488 F0 CD A7 E5 DD 21 4A F0 :81
E490 CD A9 E4 DD 21 57 F0 CD :6C
E498 A9 E4 DD 21 64 F0 CD A9 :55
E4A0 E4 DD 21 71 F0 CD A9 E4 :9D
E4A8 C9 FD 21 93 F0 FD 7E 00 :E5
E4B0 FE 0F CA B9 E4 FD 21 7E :10
E4B8 F0 FD 7E 01 DD 46 01 90 :20
E4C0 3C FE 00 D8 FE 03 D0 FD :E0
E4C8 7E 02 DD 46 02 90 3C FE :6F
E4D0 00 D8 FE 03 D0 FD 21 93 :5A
E4D8 F0 FD 7E 00 FE 0F CA F1 :33
E4E0 E4 CD 62 E1 CD 62 E1 CD :D1
E4E8 62 E1 CD 62 E1 CD 1A E5 :1F
E4F0 C9 21 E2 EF CD A7 ED 2A :46
E4F8 9F F0 DD 7E 0C 06 00 4F :4B
---------------------------------
SUM: AF C3 06 19 45 ED 50 80 :93

E500 09 22 9F F0 CD C6 E6 DD :10
E508 66 01 DD 6E 02 CD 76 E5 :DC
E510 2A 8A F0 DD 74 01 DD 75 :48
E518 02 C9 21 AA EF CD A7 ED :E6
E520 3A AB F0 3D 32 AB F0 CD :AC
E528 DB E2 DD 21 71 F0 DD 66 :5F
E530 01 DD 6E 02 CD 76 E5 DD :53
E538 21 64 F0 DD 66 01 DD 6E :04
E540 02 CD 76 E5 DD 21 57 F0 :6F
E548 DD 66 01 DD 6E 02 CD 76 :D4
E550 E5 DD 21 4A F0 DD 66 01 :61
E558 DD 6E 02 CD 76 E5 DD 21 :73
E560 7E F0 DD 66 01 DD 6E 02 :FF
E568 CD 76 E5 3A AB F0 FE 01 :FC
E570 D2 71 E0 C3 2A ED CD A5 :6F
E578 E8 CD C1 E8 CD 9A E5 23 :CD
---------------------------------
SUM: 78 66 B5 46 5C AC F4 F5 :CA

E580 03 CD 9A E5 0B 2B 11 28 :BE
E588 00 19 E5 60 69 19 44 4D :71
E590 E1 CD 9A E5 23 03 CD 9A :BA
E598 E5 C9 7E FE 20 CC BF E9 :BE
E5A0 7E FE 20 C4 B4 E9 C9 FD :C3
E5A8 21 7E F0 FD 66 01 FD 6E :5E
E5B0 02 CD A5 E8 CD C1 E8 7E :50
E5B8 FE 6D CA CD E5 FE 71 CA :20
E5C0 03 E6 FE 75 CA 72 E6 FE :7C
E5C8 7D CA A3 E6 C9 CD B9 ED :0C
E5D0 3A 91 F0 FE 0C DA 2A ED :B6
E5D8 16 14 3A A3 F0 3D 3D CA :3B
E5E0 E9 E5 15 C2 DD E5 C3 60 :8A
E5E8 ED 3A A3 F0 32 A4 F0 3E :BE
E5F0 00 32 A3 F0 CD C9 E8 CD :10
E5F8 20 EA 3A A4 F0 32 A3 F0 :9D
---------------------------------
SUM: 2E C2 76 E0 DE 96 44 A8 :A6

E600 C3 60 ED 3A 91 F0 3C 32 :39
E608 91 F0 CD F1 E9 D9 21 C6 :E8
E610 EF CD A7 ED 2A 9F F0 06 :0F
E618 00 0E 05 09 22 9F F0 CD :9A
E620 C6 E6 3A 91 F0 FE 05 C0 :2A
E628 DD 21 64 F0 DD 66 01 DD :73
E630 6E 02 CD 41 E6 CD A5 E8 :BE
E638 3E 7D CD 7C E9 CD 01 EA :A5
E640 C9 22 46 F0 CD A5 E8 CD :48
E648 C1 E8 7E FE 20 C2 6D E6 :5A
E650 23 7E FE 20 C2 6D E6 2B :FF
E658 01 28 00 09 7E FE 20 C2 :90
E660 6D E6 23 7E FE 20 C2 6D :41
E668 E6 2A 46 F0 C9 26 0E 2E :71
E670 09 C9 3A 92 F0 FE 01 D8 :65
E678 3D 32 92 F0 CD F8 E2 DD :75
---------------------------------
SUM: D9 6C 95 66 13 13 F7 2A :87

E680 21 7E F0 DD 66 01 DD 6E :1E
E688 02 3E 0F DD 21 93 F0 DD :AD
E690 74 01 DD 75 02 DD 77 00 :1D
E698 3E 00 DD 77 03 3E 1E 32 :23
E6A0 AD F0 C9 3A 92 F0 3C 32 :90
E6A8 92 F0 CD F1 E9 D9 2A 9F :CB
E6B0 F0 06 00 0E 32 09 22 9F :00
E6B8 F0 CD C6 E6 CD F8 E2 21 :31
E6C0 72 EF CD A7 ED C9 2A 9F :54
E6C8 F0 01 10 27 CD 36 E7 32 :44
E6D0 AA F0 01 E8 03 CD 36 E7 :70
E6D8 32 A9 F0 01 64 00 CD 36 :33
E6E0 E7 32 A8 F0 01 0A 00 CD :89
E6E8 36 E7 32 A7 F0 7D C6 30 :59
E6F0 32 A6 F0 26 21 2E 0C CD :16
E6F8 10 E7 2A A1 F0 ED 4B 9F :89
---------------------------------
SUM: 91 9F D7 DA 29 E7 FD 65 :53

E700 F0 ED 42 D0 ED 43 A1 F0 :B0
E708 26 21 2E 08 CD 10 E7 C9 :0A
E710 CD A5 E8 3A AA F0 CD C1 :BC
E718 E9 03 3A A9 F0 CD C1 E9 :36
E720 03 3A A8 F0 CD C1 E9 03 :4F
E728 3A A7 F0 CD C1 E9 03 3A :85
E730 A6 F0 CD C1 E9 C9 AF ED :72
E738 42 DA 40 E7 3C C3 37 E7 :60
E740 09 C6 30 C9 DD 7E 00 FE :21
E748 00 CA D5 E7 FE 01 CA 80 :CF
E750 E7 FE 02 CA AB E7 DD 66 :86
E758 01 DD 6E 02 25 CD 1B E8 :43
E760 C8 2A 3E F0 2C CD 1B E8 :1C
E768 C8 2A 3E F0 2D CD 28 E8 :2A
E770 DD 66 01 DD 6E 02 24 24 :D9
E778 CD 00 E8 2C CD 00 E8 C9 :5F
---------------------------------
SUM: 1C 86 11 85 46 15 F9 FD :89

E780 DD 66 01 DD 6E 02 24 24 :D9
E788 CD 1B E8 C8 2A 3E F0 2C :1C
E790 CD 1B E8 C8 2A 3E F0 25 :15
E798 2D CD 28 E8 DD 66 01 DD :2B
E7A0 6E 02 25 CD 00 E8 2C CD :43
E7A8 00 E8 C9 DD 66 01 DD 6E :40
E7B0 02 2D CD 1B E8 C8 2A 3E :2F
E7B8 F0 24 CD 1B E8 C8 2A 3E :14
E7C0 F0 25 CD 28 E8 DD 66 01 :36
E7C8 DD 6E 02 2C 2C CD 00 E8 :5A
E7D0 24 CD 00 E8 C9 DD 66 01 :E6
E7D8 DD 6E 02 2C 2C CD 1B E8 :75
E7E0 C8 2A 3E F0 24 CD 1B E8 :14
E7E8 C8 2A 3E F0 25 2D CD 28 :67
E7F0 E8 DD 66 01 DD 6E 02 2D :A6
E7F8 CD 00 E8 24 CD 00 E8 C9 :57
---------------------------------
SUM: 17 A3 1C A2 D1 19 1B E1 :5E

E800 22 3E F0 CD A5 E8 CD C1 :38
E808 E8 7E FE 20 CC BF E9 7E :76
E810 FE 20 C4 B4 E9 2A 3E F0 :D7
E818 F6 FF C9 22 3E F0 CD A5 :80
E820 E8 CD C1 E8 7E FE 23 C9 :C6
E828 DD 74 01 DD 75 02 CD A5 :18
E830 E8 DD 7E 03 FE 01 C2 42 :49
E838 E8 CD 4B E8 3E 02 DD 77 :7C
E840 03 C9 CD 6D E8 3E 01 DD :0A
E848 77 03 C9 DD 7E 04 CD B4 :23
E850 E9 03 DD 7E 05 CD B4 E9 :B6
E858 0B 21 28 00 09 44 4D DD :CB
E860 7E 06 CD B4 E9 03 DD 7E :4C
E868 07 CD B4 E9 C9 DD 7E 08 :9D
E870 CD B4 E9 03 DD 7E 09 CD :9E
E878 B4 E9 0B 21 28 00 09 44 :3E
---------------------------------
SUM: 07 26 16 FC F2 75 8C E9 :1B

E880 4D DD 7E 0A CD B4 E9 03 :1F
E888 DD 7E 0B CD B4 E9 C9 2A :C3
E890 44 F0 54 5D 29 29 19 11 :61
E898 82 35 19 22 44 F0 7C E6 :88
E8A0 03 32 42 F0 C9 22 48 F0 :8A
E8A8 26 00 29 44 4D 29 29 09 :3B
E8B0 29 29 01 00 30 09 ED 4B :C4
E8B8 48 F0 48 06 00 09 44 4D :20
E8C0 C9 78 69 F6 D0 CB AF 67 :51
E8C8 C9 CD CC E9 CD 41 EC 01 :46
E8D0 2A 30 1E 0B 16 07 7E CB :E9
E8D8 3F CB 3F CB 3F CB 3F CD :2A
E8E0 09 E9 03 03 7E E6 0F CD :38
E8E8 09 E9 03 03 23 15 C2 D6 :C8
E8F0 E8 E5 60 69 01 1C 00 ED :A0
E8F8 42 44 4D E1 E5 21 50 00 :0A
---------------------------------
SUM: C1 06 EF 95 AD 29 62 45 :C8

E900 09 44 4D E1 1D C2 D4 E8 :16
E908 C9 FE 01 C2 14 E9 3E 69 :2E
E910 CD 7C E9 C9 FE 02 C2 1D :DA
E918 E9 CD 53 E9 C9 FE 03 C2 :7E
E920 2B E9 3E 6D CD 7C E9 CD :BE
E928 01 EA C9 FE 04 C2 39 E9 :9A
E930 3E 71 CD 7C E9 CD 01 EA :99
E938 C9 FE 05 C2 47 E9 3E 75 :71
E940 CD 7C E9 CD 01 EA C9 FE :B1
E948 06 C0 3E 79 CD 7C E9 CD :7C
E950 01 EA C9 CD E5 E9 CD BF :DB
E958 E9 03 CD BF E9 0B E5 21 :72
E960 28 00 09 44 4D E1 CD BF :2F
E968 E9 03 CD BF E9 E5 60 69 :0F
E970 01 28 00 ED 42 44 4D 0B :F4
E978 E1 E6 00 C9 32 90 F0 CD :0F
---------------------------------
SUM: 6B 07 F6 89 3F 93 06 F0 :B9

E980 B4 E9 03 3A 90 F0 3C CD :63
E988 B4 E9 0B E5 21 28 00 09 :DF
E990 44 4D E1 3A 90 F0 3C 3C :A4
E998 CD B4 E9 03 3A 90 F0 3C :63
E9A0 3C 3C CD B4 E9 E5 60 69 :90
E9A8 01 28 00 ED 42 44 4D 0B :F4
E9B0 E1 C9 3E 23 ED 79 CB A0 :DC
E9B8 3E 27 ED 79 CB E0 C9 3E :7D
E9C0 20 ED 79 CB A0 3E 07 ED :23
E9C8 79 CB E0 C9 21 00 D0 22 :00
E9D0 37 F0 01 E7 D3 2A 37 F0 :33
E9D8 3E 23 77 23 22 37 F0 ED :31
E9E0 42 C2 D5 E9 C9 ED 43 3B :F6
E9E8 F0 D9 ED 4B 3B F0 CD C1 :BA
E9F0 E8 36 20 23 36 20 01 28 :E0
E9F8 00 09 36 20 2B 36 20 D9 :B9
---------------------------------
SUM: FD CC B9 AE 79 EC D8 89 :F6

EA00 C9 ED 43 3B F0 D9 ED 4B :35
EA08 3B F0 CD C1 E8 3A 90 F0 :5B
EA10 77 3C 23 77 01 28 00 09 :7F
EA18 2B 3C 77 23 3C 77 D9 C9 :56
EA20 3E 03 32 B0 F0 26 09 2E :70
EA28 04 CD A5 E8 21 10 F1 16 :96
EA30 0F 7E CD C1 E9 23 03 15 :3F
EA38 C2 31 EA DD 21 71 F0 06 :42
EA40 0C DD 70 02 3E 07 DD 77 :F4
EA48 01 CD AB E7 DD 21 64 F0 :B2
EA50 3E 0B 06 0C DD 77 01 DD :8D
EA58 70 02 CD AB E7 DD 21 57 :26
EA60 F0 3E 0F 06 0C DD 77 01 :A4
EA68 DD 70 02 CD AB E7 DD 21 :AC
EA70 4A F0 3E 13 06 0C DD 77 :F1
EA78 01 DD 70 02 CD AB E7 DD :8C
---------------------------------
SUM: 8C 06 E5 54 99 73 BE 7D :12

EA80 21 93 F0 3E 17 06 0C DD :E8
EA88 77 01 DD 70 02 CD AB E7 :26
EA90 DD 21 7E F0 3E 07 DD 77 :05
EA98 01 3E 0E DD 77 02 CD 80 :F0
EAA0 E7 26 06 2E 0E CD A5 E8 :A9
EAA8 CD C1 E8 3E 23 77 26 19 :8D
EAB0 2E 0E CD A5 E8 CD C1 E8 :0C
EAB8 3E 23 77 26 07 2E 0B CD :0B
EAC0 A8 EB 26 0B 2E 0B CD A8 :72
EAC8 EB 26 0F 2E 0B CD A8 EB :B9
EAD0 26 13 2E 0B CD A8 EB 26 :F8
EAD8 17 2E 0B CD A8 EB DD 21 :AE
EAE0 7E F0 CD 62 E1 01 00 1C :9B
EAE8 3E 0E ED 79 05 ED 78 32 :4E
EAF0 8C F0 E6 04 CC 56 E7 3A :A9
EAF8 8C F0 E6 08 CC 80 E7 3A :D7
---------------------------------
SUM: 3A 3B 7F AA 1A 4A 7B 0D :8A

EB00 8C F0 E6 20 CA 20 EB 3A :91
EB08 8C F0 E6 04 21 3C EF CC :7E
EB10 A7 ED 3A 8C F0 E6 08 21 :59
EB18 3C EF CC A7 ED C3 E2 EA :1A
EB20 DD 21 71 F0 CD 82 EB DD :76
EB28 21 64 F0 CD 82 EB DD 21 :AD
EB30 57 F0 CD 82 EB DD 21 4A :C9
EB38 F0 CD 82 EB DD 21 93 F0 :AB
EB40 CD 82 EB DD 21 7E F0 CD :73
EB48 82 EB CD EA EB DD 21 7E :8B
EB50 F0 CD 95 EB DD 21 93 F0 :BE
EB58 CD 95 EB DD 21 4A F0 CD :52
EB60 95 EB DD 21 57 F0 CD 95 :27
EB68 EB DD 21 64 F0 CD 95 EB :8A
EB70 DD 21 71 F0 CD 95 EB 3A :E6
```

```
EB78 B0 F0 3D 32 B0 F0 C8 C3 :3A
--------------------------------
SUM: 59 A6 66 B7 AD 78 E9 CE :F8

EB80 BB EA CD AB E7 CD 62 E1 :14
EB88 CD AB E7 CD 62 E1 CD AB :E7
EB90 E7 CD 62 E1 C9 CD D5 E7 :49
EB98 CD 62 E1 CD D5 E7 CD 62 :C8
EBA0 E1 CD D5 E7 CD 62 E1 C9 :43
EBA8 E5 CD 8F E8 E1 FE 00 06 :0E
EBB0 71 CA CD EB FE 01 06 7D :75
EBB8 CA CD EB FE 02 06 61 CA :B3
EBC0 CD EB CD A5 E8 CD C1 E8 :88
EBC8 CD F1 E9 D9 C9 78 32 42 :35
EBD0 F0 CD A5 E8 CD C1 E8 3A :FA
EBD8 42 F0 77 3C 23 77 06 00 :85
EBE0 0E 28 09 3C 3C 77 2B 3D :96
EBE8 77 C9 CD C1 E8 DD 21 7E :32
EBF0 F0 DD 66 01 DD 6E 02 CD :4E
EBF8 A5 E8 CD C1 E8 7E FE 71 :F0
--------------------------------
SUM: 23 44 EE 3F 1F 86 46 48 :C7

EC00 CA 0E EC FE 7D CA 22 EC :17
EC08 FE 61 CA 33 EC C9 21 C6 :F8
EC10 EF CD A7 ED 2A 9F F0 01 :0A
EC18 E8 03 09 22 9F F0 CD C6 :38
EC20 E6 C9 21 72 EF CD A7 ED :92
EC28 3A 92 F0 3C 32 92 F0 CD :79
EC30 F8 E2 C9 CD B9 ED 3A AB :FB
EC38 F0 3C 32 AB F0 CD DB E2 :83
EC40 C9 3A A3 F0 FE 00 CA 81 :DF
EC48 EC FE 01 CA 8A EC FE 02 :2B
EC50 CA 9A EC FE 03 CA AA EC :B1
EC58 FE 04 CA BA EC FE 05 CA :3F
EC60 CA EC FE 06 CA DA EC FE :48
EC68 07 CA EA EC FE 08 CA FA :71
EC70 EC FE 09 CA 0A ED FE 0A :BC
EC78 CA 1A ED FE 0B CA D1 ED :62
--------------------------------
SUM: AB 5C AA 92 50 88 A8 E8 :AB

EC80 C9 3E D0 32 A5 F0 21 20 :DF
EC88 F1 C9 3E F0 32 A5 F0 26 :D5
EC90 1A 2E 15 22 8A F0 21 6D :87
EC98 F1 C9 3E F0 32 A5 F0 26 :D5
ECA0 06 2E 01 22 8A F0 21 BA :AC
ECA8 F1 C9 3E F0 32 A5 F0 26 :D5
ECB0 16 2E 01 22 8A F0 21 07 :09
ECB8 F2 C9 3E F0 32 A5 F0 26 :D6
ECC0 0A 2E 0D 22 8A F0 21 54 :56
ECC8 F2 C9 3E D0 32 A5 F0 26 :B6
ECD0 10 2E 05 22 8A F0 21 A1 :A1
ECD8 F2 C9 3E D0 32 A5 F0 26 :B6
ECE0 0C 2E 0D 22 8A F0 21 EE :F2
ECE8 F2 C9 3E C0 32 A5 F0 26 :A6
ECF0 08 2E 05 22 8A F0 21 3B :33
ECF8 F3 C9 3E CC 32 A5 F0 26 :B3
--------------------------------
SUM: BB C8 FB 0C FB A8 88 9C :51

ED00 0E 2E 13 22 8A F0 21 88 :94
ED08 F3 C9 3E B0 32 A5 F0 26 :97
ED10 1C 2E 09 22 8A F0 21 D5 :E5
ED18 F3 C9 3E B0 32 A5 F0 26 :97
ED20 16 2E 0D 22 8A F0 21 22 :30
ED28 F4 C9 3E 00 32 A3 F0 CD :8D
ED30 C9 E8 26 0A 2E 0C CD A5 :8D
ED38 E8 CD C1 E8 06 0A 11 E9 :68
ED40 F0 CD 72 E2 3E F0 32 A5 :16
ED48 F0 DD 21 93 F0 3E 17 DD :A3
ED50 77 01 3E 0B DD 77 02 CD :E4
ED58 5F E2 CD 62 E1 C3 3B E0 :2F
ED60 3A A3 F0 F5 3E 00 32 A3 :D5
ED68 F0 CD C9 E8 F1 3C 32 A3 :70
ED70 F0 26 0A 2E 0C CD A5 E8 :B4
ED78 CD C1 E8 06 0A 11 F3 F0 :7A
--------------------------------
SUM: 68 7E 13 AB 99 55 93 73 :98

ED80 CD 72 E2 3E F0 32 A5 F0 :16
ED88 DD 21 93 F0 3E 17 DD 77 :2A
ED90 01 3E 0B DD 77 02 CD 5F :CC
ED98 E2 CD 62 E1 3E 00 32 91 :F3
EDA0 F0 CD C9 E8 C3 71 E0 16 :98
EDA8 1C 7E 01 00 1C ED 79 05 :22
EDB0 23 7E ED 79 15 C2 A9 ED :74
EDB8 C9 21 03 F0 3E 01 77 E5 :78
EDC0 F5 21 FE EF CD A7 ED F1 :55
EDC8 E1 3C 3C FE FD DA BE ED :D9
EDD0 C9 3E 00 32 A3 F0 CD C9 :62
EDD8 E8 FD 21 14 EF 3A A3 F0 :D6
EDE0 3C FE 09 CA AD EE 32 A3 :7D
EDE8 F0 DD 21 4A F0 26 02 2E :7E
EDF0 04 DD 74 01 DD 75 02 DD :87
EDF8 21 57 F0 2E 06 DD 74 01 :EE
--------------------------------
SUM: 5D 2F 85 B3 F1 7D BF 8A :7B

EE00 DD 75 02 DD 21 64 F0 2E :D4
EE08 08 DD 74 01 DD 75 02 DD :8B
EE10 21 71 F0 2E 0A DD 74 01 :0C
EE18 DD 75 02 DD 21 93 F0 2E :03
EE20 0C DD 74 01 DD 75 02 21 :D3
EE28 0F EF 06 05 FD 7E 00 77 :FB
EE30 FD 23 23 10 F7 06 05 21 :76
EE38 0F EF 7E FE 00 C2 46 EE :70
EE40 23 10 F7 C3 DD ED CD 4C :D0
EE48 EE C3 35 EE CD 62 E1 3A :1E
EE50 0F EF FE 00 CA 62 EE 3D :53
EE58 32 0F EF DD 21 4A F0 CD :35
EE60 80 E7 3A 10 EF FE 00 CA :68
EE68 75 EE 3D 32 10 EF DD 21 :CF
EE70 57 F0 CD 80 E7 3A 11 EF :B5
EE78 FE 00 CA 88 EE 3D 32 11 :BE
--------------------------------
SUM: A6 AC AA D5 63 63 4F 5C :42

EE80 EF DD 21 64 F0 CD 80 E7 :75
EE88 3A 12 EF FE 00 CA 9B EE :8C
EE90 3D 32 12 EF DD 21 71 F0 :CF
EE98 CD 80 E7 3A 13 EF FE 00 :6E
EEA0 C8 3D 32 13 EF DD 21 93 :CA
EEA8 F0 CD 80 E7 C9 26 0B 2E :4C
EEB0 11 CD A5 E8 21 B1 F0 3E :6B
EEB8 0A 32 A3 F0 CD F1 EE 26 :A1
EEC0 0A 2E 12 CD A5 E8 21 BB :80
EEC8 F0 3E 0C 32 A3 F0 CD F1 :BD
EED0 EE 26 08 2E 13 CD A5 E8 :B7
EED8 21 FD F0 3E 13 32 A3 F0 :24
EEE0 CD F1 EE CD A9 E2 3E 01 :43
EEE8 32 A3 F0 CD CC E9 C3 8A :94
EEF0 EC 7E CD C1 E9 C5 E5 21 :AC
EEF8 3C EF CD A7 ED CD 62 E1 :9C
--------------------------------
SUM: 36 3A 91 CA 3F 80 12 FB :97

EF00 E1 C1 23 03 3A A3 F0 3D :D2
EF08 C8 32 A3 F0 C3 F1 EE 00 :2F
EF10 00 00 00 00 17 19 1A 19 :63
EF18 17 15 15 15 15 15 11 11 :A2
EF20 11 11 11 00 0D 0E 0F 00 :5D
EF28 0B 0B 0B 0B 0B 07 00 00 :3E
EF30 00 07 05 00 05 00 05 03 :19
EF38 03 03 03 03 00 96 01 00 :A3
EF40 02 82 00 78 05 00 06 00 :07
EF48 07 3F 08 10 09 10 0A 10 :91
EF50 0B 00 0C 05 0D 00 00 00 :29
EF58 01 00 02 00 03 00 04 1E :28
EF60 05 00 06 00 07 3C 08 10 :66
EF68 09 10 0A 10 0B 00 0C 05 :4F
EF70 0D 00 00 00 01 00 02 21 :31
EF78 03 00 04 00 05 00 06 00 :12
--------------------------------
SUM: 12 FF 29 B3 7C B9 4E CE :3E

EF80 07 38 08 00 09 10 0A 00 :6A
EF88 0B 00 0C 05 0D 00 00 BD :E6
EF90 01 00 02 00 03 00 04 00 :0A
EF98 05 00 06 00 07 38 08 10 :62
EFA0 09 10 0A 10 0B 00 0C 03 :4D
EFA8 0D 08 00 BD 01 01 02 00 :D6
EFB0 03 00 04 00 05 00 06 00 :12
EFB8 07 38 08 10 09 10 0A 10 :8A
EFC0 0B 00 0C 05 0D 05 00 00 :2E
EFC8 01 00 02 00 03 00 04 14 :1E
EFD0 05 00 06 0A 07 29 08 00 :4D
EFD8 09 10 0A 10 0B 00 0C 04 :4E
EFE0 0D 04 00 00 01 64 02 00 :78
EFE8 03 00 04 96 05 00 06 1F :C7
EFF0 07 1F 08 10 09 10 0A 10 :71
EFF8 0B 00 0C 1A 0D 53 00 00 :91
--------------------------------
SUM: 74 BB 68 C1 78 4E 5E 27 :A3

F000 01 00 02 00 03 00 04 00 :0A
F008 05 00 06 00 07 38 08 00 :52
F010 09 10 0A 00 0B 00 0C 04 :3E
F018 0D 00 00 00 01 00 02 00 :10
F020 03 00 04 00 05 00 06 00 :12
F028 07 3F 08 00 09 00 0A 00 :61
F030 0B 00 0C 00 0D 00 00 00 :24
F038 00 00 00 00 00 00 00 00 :00
F040 00 00 00 00 00 00 00 00 :00
F048 00 00 00 00 00 00 00 00 :00
F050 00 00 00 00 00 00 00 00 :00
F058 00 00 00 00 00 00 00 00 :00
F060 00 00 00 00 00 00 00 00 ;00
F068 00 00 00 00 00 00 00 00 :00
F070 00 00 00 00 00 00 00 00 :00
F078 00 00 00 00 00 00 00 00 :00
--------------------------------
SUM: 31 4F 2A 00 31 38 2A 04 :41

F080 00 00 00 00 00 00 00 00 :00
F088 00 00 00 00 00 00 00 00 :00
F090 00 00 00 00 00 00 00 00 :00
F098 00 00 00 00 00 00 00 00 :00
F0A0 00 00 00 00 00 00 00 00 :00
F0A8 00 00 00 00 00 00 00 00 :00
F0B0 00 CF BC DD BA DE 20 B6 :D6
F0B8 B8 C3 D9 69 6E 20 5A 38 :DD
F0C0 30 27 73 20 42 61 72 48 :47
F0C8 49 4B 41 4C 20 20 28 B1 :3A
F0D0 C5 C0 29 4A 55 4E 4A 49 :2E
F0D8 43 48 4F 55 52 4F 55 4D :72
F0E0 45 41 52 59 5A 45 4E 4A :68
F0E8 49 47 41 4D 45 20 20 4F :F2
F0F0 56 45 52 20 3E 3E 43 4C :18
F0F8 45 41 52 3C 3C CA A4 B4 :72
--------------------------------
SUM: 62 1A F8 53 4A 89 08 16 :B8

F100 B2 B4 DD C6 20 CC D2 C2 :89
F108 C3 DE B1 D9 A5 A5 A5 A1 :BB
F110 2A 20 CE DE B0 C5 BD 20 :48
F118 BD C3 B0 BC DE 20 2A 20 :34
F120 22 22 22 22 22 22 22 22 :10
F128 22 22 22 22 22 22 22 22 :10
F130 22 22 22 22 22 22 22 22 :10
F138 22 22 22 22 22 22 22 22 :10
F140 22 22 22 22 22 22 22 22 :10
F148 22 22 22 22 22 22 22 22 :10
F150 22 22 22 22 22 22 22 22 :10
F158 22 22 22 22 22 22 22 22 :10
F160 22 22 22 22 22 22 22 22 :10
F168 22 22 22 22 22 22 24 11 :01
F170 15 11 42 22 21 12 22 14 :F3
F178 22 21 12 21 24 12 22 11 :DF
--------------------------------
SUM: E7 FB B4 D0 EC CE F8 0B :23

F180 22 12 22 21 11 21 11 12 :CC
F188 24 12 42 22 22 12 24 21 :13
F190 32 11 41 22 22 11 21 12 :0C
F198 11 11 21 12 22 21 22 21 :DB
F1A0 14 22 11 12 22 21 22 12 :D0
F1A8 12 11 22 14 21 12 22 42 :F0
F1B0 22 21 12 24 22 11 51 11 :0E
F1B8 22 62 22 62 11 12 42 12 :7F
F1C0 22 21 12 15 22 12 12 12 :C2
F1C8 42 22 22 21 12 42 14 12 :21
F1D0 11 21 21 12 11 12 12 22 :BC
F1D8 41 22 42 12 22 42 11 22 :4E
F1E0 21 12 22 11 21 11 21 11 :CA
F1E8 11 12 22 21 22 22 23 12 :DF
F1F0 42 12 42 21 11 22 22 12 :1E
F1F8 22 21 22 21 21 21 11 14 :ED
--------------------------------
SUM: 3F D9 6C F1 C9 D9 0F 8E :B4

F200 21 11 24 21 22 42 22 42 :3F
F208 21 11 11 22 62 24 21 22 :2E
F210 12 24 21 11 12 21 12 12 :BF
F218 11 11 22 22 41 22 42 21 :2C
F220 22 41 12 22 21 11 22 21 :0C
F228 11 42 21 22 22 21 42 21 :3C
F230 21 21 11 21 11 21 22 21 :E9
F238 24 22 22 15 21 21 22 11 :F2
F240 21 12 11 24 21 12 22 31 :EE
F248 52 12 21 22 12 21 11 12 :FD
F250 42 11 12 24 42 21 22 21 :2F
F258 11 22 24 21 24 21 22 11 :F0
F260 21 12 22 12 15 22 22 22 :E2
F268 12 12 22 22 21 12 14 22 :D1
F270 24 11 41 23 12 41 21 21 :2E
F278 22 22 22 11 22 22 22 21 :FE
--------------------------------
SUM: 1C CB ED E3 4F 29 2F 06 :64

F280 62 12 12 21 12 21 22 12 :0E
F288 24 22 12 22 21 12 22 12 :E1
F290 12 22 12 22 22 11 22 21 :DE
F298 21 12 41 12 42 21 24 22 :2F
F2A0 24 22 22 22 24 14 22 22 :06
F2A8 41 12 11 21 11 21 12 22 :EB
F2B0 22 24 26 22 21 42 21 12 :24
F2B8 11 12 14 22 21 22 12 41 :EF
F2C0 22 11 21 11 12 22 12 21 :CC
F2C8 11 22 11 11 12 22 13 11 :AD
F2D0 12 11 22 41 21 12 11 44 :0E
F2D8 11 21 22 22 12 12 42 21 :FD
F2E0 21 12 12 22 22 11 24 22 :E0
F2E8 22 11 11 51 11 12 22 24 :FE
F2F0 11 42 22 24 22 21 12 11 :FF
F2F8 21 12 11 12 21 12 22 21 :CC
--------------------------------
SUM: 1C AE B0 2C DB BC E3 0D :2D

F300 12 12 24 42 22 11 22 22 :01
F308 12 11 11 24 11 21 12 22 :BE
F310 51 11 22 22 21 14 12 22 :0F
F318 42 21 16 22 22 11 12 21 :01
F320 21 12 11 22 22 22 21 42 :0D
F328 22 11 21 12 11 41 11 12 :DB
F330 24 21 12 13 22 22 22 11 :E1
F338 22 24 22 22 21 42 41 22 :50
F340 22 22 41 21 21 21 21 41 :4A
F348 12 21 26 22 21 21 25 14 :F6
F350 21 21 11 22 21 12 12 22 :DC
F358 22 21 21 22 22 22 12 11 :ED
F360 21 21 11 21 11 12 12 41 :EA
F368 22 41 22 41 12 12 12 21 :1D
F370 21 21 21 22 12 42 11 22 :0C
F378 21 23 41 22 12 21 21 21 :1C
--------------------------------
SUM: 3C E8 01 40 B8 1B AD 3B :20

F380 21 22 21 21 24 21 24 22 :10
F388 42 22 21 11 22 22 24 21 :1F
F390 21 22 12 21 21 12 21 42 :0C
F398 14 22 12 22 12 22 12 21 :D1
F3A0 21 22 12 22 12 21 22 52 :1E
F3A8 21 14 11 11 22 21 21 22 :DD
F3B0 12 22 42 22 12 22 12 21 :FF
F3B8 12 21 31 22 12 21 24 12 :EF
F3C0 21 12 41 11 22 12 22 21 :FC
F3C8 22 11 61 24 21 12 42 22 :4F
F3D0 22 42 21 22 24 22 21 11 :1F
F3D8 24 12 22 42 21 22 12 22 :11
F3E0 12 12 12 41 12 22 12 22 :DF
F3E8 12 22 21 22 12 11 21 42 :FD
F3F0 11 22 21 12 22 22 22 26 :F2
F3F8 11 21 22 12 12 12 14 22 :C0
--------------------------------
SUM: CD EF 57 0C B1 CB F4 6F :FE

F400 22 21 12 24 22 22 21 11 :EF
F408 21 11 21 21 21 22 41 22 :1A
F410 22 22 42 24 25 21 12 12 :14
F418 12 12 12 22 13 24 14 22 :C5
F420 24 22 22 22 41 55 14 22 :56
F428 22 41 41 21 22 12 12 14 :1F
F430 22 22 24 22 22 22 22 21 :11
F438 21 12 11 11 21 12 22 21 :CB
F440 12 22 21 22 22 21 22 22 :FE
F448 11 22 22 12 42 21 12 31 :0D
F450 21 62 24 11 21 22 22 21 :3E
F458 12 11 22 22 21 22 22 22 :EE
F460 22 21 21 21 21 21 21 12 :FA
F468 24 22 22 41 42 22 42 00 :4F
F470 00                      :00
--------------------------------
SUM: 9C F7 EB CA 2A ED CD 87 :B3
```

X68000用
# うれしい!たのしい!大好き!
Takahashi Tomoyuki 高橋 呂志

X1/turbo用
# SPANISH BLUE
Tanaka Kazunari 田中 一成

今月は，芸術の秋とばかりに2大メジャーグループのぶつかり合いとなりました。どちらも力作，ぜひ打ち込んでみてください。また，今月は，久し振りにX1用プログラムの登場です。アルバムからの選曲なので知らない人もいるかもしれませんが，聴けば「ああ，TM」と納得することでしょう。

## うれしい！たのしい！初登場！

X68000OPMD用には，Dreams Come Trueの「うれしい！ たのしい！ 大好き！」をお届けしましょう。この曲は，約1年前のアルバム「LOVE GOES ON…」の中からの選曲です。ドリカムといえば，「うれしはずかし朝帰り」なども有名ですし，CMソングでも何曲か使われているので聴いたこともあるでしょう。女の人が1人と，男の人が2人の3人組です。

さて，作品は初投稿の高橋君によるものです。さすがにちょっと煮詰め不足っぽいのですが，期待票も込めて晴れて掲載となりました。もちろん，掲載される以上，箸にも棒にも引っ掛からないような作品ではありません。今後の参考のために，いくつか気づいた点を指摘しておきましょう。

全体的にちょっと音色のツメがあまいようです。間奏のブラスが弱めなのはFM音源の宿命的なものですが，ヴォーカルはもっと力強い音にできると思います。なんといっても，ドリカムの売りは吉田美和さんのヴォーカルなのですから，もっと研究するとよいでしょう。いまのままでは，バッキングにヴォーカルが埋もれ気味ですよね。バッキングと似ている音を使わないだけでも，ずいぶんと雰囲気がかわると思います。試してみてください。

Dreams Come True

それから，フィニッシュがイマイチではないでしょうか。フェードアウトがバラバラになっています。めりはりをつけてアレンジしてみるといいかもしれません。たとえば，ループを「うれしいたのしいだいすき」で切ってしまうなんてのも有効でしょう。

リストの入力について注意点があります。各トラックの最後のあたりに，ひとつおきに空白がある文字列がありますが，正確に入力してください。

高橋君は就職先を探しているそうですね。もう決まっているのかな？ 就職は「くやしい！ かなしい！ 大嫌い！」ではなく，「うれしい！ たのしい！ 大好き！」になっているといいですね。仕事をはじめるとなかなかパソコンをいじる暇がなくなるようですが，もっともっと研究して常連目指してがんばってくださいね。

## マイナーチェンジ後の初登場

X1のMusicBASICには，いまやTMNとなったTMネットワークの「SPANISH BLUE」をお送りしましょう。アルバム「DRESS」からの選曲になります。このアルバムは，約1年半ぐらい前の作品ですね。ちなみに，この時期ではまだTMネットワークを名乗っていたはずです。

このプログラムでは，音色設定に拡張プログラム方式を使っていますので，そちらのほうを準備してください。あの「ねこバス」などのやつですね。プログラム中でロードしている“Voice set r.Bin”というのがそれにあたりますので，ファイルネームは自分で変更してください。

TMN

FM音源8声＋PSGを使っていますので，ミキシングには気をつけてください。実際にPSGから出ている音はハイハットのみですので，好みの大きさにしてかまわないと思います。

この作品も音色で苦労しているようですね。もともとTMネットワーク自体が音色に頼った曲作りをしているようなところがありますので，FM音源＋PSGだけでは役不足になるのは目に見えていると思います。できるだけメロディアスでコード進行がきれいだったり，サンプリングマシンに頼りすぎていない曲のほうが仕上がりがよくなりますので，次回はそういった曲を選ぶとよいかもしれません。特にPCMのないX1では，かなり苦労してもきれいに仕上げるのは至難の技ですからね。

難しい曲を選んでしまったわりには，作品のデキ具合はなかなかですね。今後の田中君に期待します。なんといってもX1の投稿は少ないので，掲載レベルの作品を送ってくれる人は貴重な存在なんですよ。X1の毎月掲載のためにも，精進してバシバシ投稿してください。 (S.K.)

日本音楽著作権協会(出)許諾第9171172-101号

## リスト1 うれしい!たのしい!大好き!

```
10 /*□■□■□■□■□■□■□■□■□■□■□■□■□■□
20 /*■                                              ■
30 /*□         ♪ うれしい!たのしい!大好き! ♪         □
40 /*■                                              ■
50 /*□              作詩・作曲  吉田美和               □
60 /*■                                              ■
70 /*□       LOVE GOES ON･･･   Dreams Come True      □
80 /*■                                              ■
90 /*□         Presented by Tomoyuki Takahashi       □
100 /*■                                             ■
110 /*□■□■□■□■□■□■□■□■□■□■□■□■□■□
120 /*
130 char v(4,10)
140 v={ 44,15, 0, 0, 0, 0, 0, 0, 0, 3, 0, /*コーラス
150      31, 0, 0, 0, 0,21, 0, 8, 7, 0, 0,
160      31, 5,10, 6, 2, 0, 0, 8, 7, 0, 0,
170      31, 0, 0, 0, 0,20, 0, 8, 3, 0, 0,
180      31, 5,10, 6, 2, 0, 0, 8, 3, 0, 0 }
190 m_vset(88,v) /*----------------------------------------
200 /*
210 v={  4,15, 0, 0, 0, 0, 0, 0, 0, 3, 0, /*C&G
220      23, 3, 2, 0, 5,25, 1, 2, 7, 0, 0,
230      15, 5, 4, 9, 6, 0, 1, 2, 3, 0, 0,
240      23, 3, 2, 0, 5,25, 1, 3, 3, 0, 0,
250      15, 5, 4, 9, 6, 0, 1, 3, 7, 0, 0 }
260 m_vset(180,v)/*----------------------------------------
270 /*
280 v={ 60,15, 0, 0, 0, 0, 0, 0, 0, 3, 0,
290      31, 0, 0, 0, 0,32, 0, 4, 3, 0, 0,
300      18, 6, 2, 7, 2, 2, 0, 8, 3, 0, 0,
310      31, 6, 2, 7, 2, 2, 0, 4, 7, 0, 0,
320      31, 6, 2, 7, 2, 2, 0, 4, 7, 0, 0 }
330 m_vset(86,v) /*----------------------------------------
340 /*
350 v={ 58,15, 0, 0, 0, 0, 0, 0, 0, 3,83,
360      30, 1, 0, 5, 1,35, 1, 1, 2, 0, 1,
370      31, 1, 0, 5, 1,25, 1, 2, 1, 0, 1,
380      30, 1, 0, 5, 1,44, 1, 2, 1, 0, 1,
390      12, 2, 0, 7, 0, 2, 1, 1, 3, 0, 1 }
400   m_vset(84,v) /*----------------------------------------
410 v={ 60,15, 0, 0, 0, 0, 0, 0, 0, 2, 0, /*HI-HAT CLOSE
420      31, 6, 6, 0, 6, 0, 0,15, 7, 1, 0,
430      31,16, 4,15,15,10, 0, 2, 3, 2, 0,
440      31,18, 6, 3, 4, 0, 0, 1, 7, 2, 0,
450      31,20, 6,15,12, 5, 2, 7, 3, 3, 0 }
460 m_vset(71,v) /*----------------------------------------
470 /*
480 v={ 60,15, 0, 0, 0, 0, 0, 0, 0, 2, 0, /*HI-HAT OPEN
490      31, 6, 6, 0, 6, 0, 0,15, 7, 1, 0,
500      31,11, 4, 6, 8,13, 0, 2, 3, 2, 0,
510      31,18, 6, 4, 4, 0, 0, 1, 7, 2, 0,
520      31,25, 6, 6,11, 8, 2, 7, 3, 3, 0 }
530 m_vset(72,v) /*----------------------------------------
540 /*
550 v={ 50,15, 0, 0, 0, 0, 0, 0, 0, 3, 0, /*Synthe BASS
560      31, 8, 3, 1, 1,28, 0, 1, 7, 0, 0,
570      28,12, 3, 1, 6,36, 0, 4, 3, 0, 0,
580      28,10, 3, 1, 5,40, 0, 0, 3, 0, 0,
590      31, 4, 5, 7, 3, 0, 0, 1, 0, 0, 0 }
600 m_vset(74,v) /*----------------------------------------
610 /*
620 v={ 61,15, 0, 0, 0, 0, 0, 0, 0, 3, 0, /*Synthe BRASS
630      24,12, 2, 5, 1,25, 0, 8, 3, 0, 0,
640      29,31, 0, 8, 0, 8, 0, 8, 7, 0, 0,
650      28,31, 0, 7, 0, 8, 0, 8, 7, 0, 0,
660      27,31, 0, 7, 0, 6, 0, 4, 7, 0, 0 }
670 m_vset(78,v) /*----------------------------------------
680 /*
690 v={ 60,15, 0, 0, 0, 0, 0, 0, 0, 3, 0,  /*Synthe ﾒﾛﾃﾞｨ
700      31, 2, 1,20, 2,28, 0, 3, 7, 0, 0,
710      31, 7, 3, 8, 2, 0, 0, 2, 7, 0, 0,
720      20, 4, 2, 1, 2,30, 0, 5, 3, 0, 0,
730      18, 7, 3, 8, 2, 0, 0, 1, 3, 0, 0}
740 m_vset(87,v) /*----------------------------------------
750 /*
760 char vol(8)={0,0,127,110,109,110,120,127,120}
770 m_init():for i=1 to 8:m_alloc(i,7000):m_assign(i,i):next
780 str m(35)[255]
790 str s1[256],s2[256],m1[256],m2[256],m20[256],m21[256],m3[2
56],m4[256],m5[256],m6[256]
800 str drm[30],drm0[30],drm1[256]
810 drm="y3,2y2,29ey2,29ey3,3" : drm0="y3,1y2,28y3,3"
820 s1="@71v13e@72e8y2,15@71ee@72e&y2,10e@71ee@72y2,10e8y2,15@
71ee@72e8
830 s2="y2,10@71ev13e"+drm0+"eey2,15eeey2,10eeey2,10eey2,15ee"
+drm
840 m1="o2r418c4e-4<c4>b-a-gf16e-.dc>b-<c.c.d&d2d4.e-&e-2 r4c4
e-4<c.>b-b-16a-gfe-dc>b-<c4.d2e-16d16e-2
850 m20="18e-4a-4<e-4dc>b-a-16g.fe-de-.e-.f&f2f4.g&g2 r4e-4a-4
<e-.dd16c>b-a-gfe-de-4.f2g16f16g2
860 m2="r4"+m20
870 m3="|:b-4e-8fe-4b-b-b-8b-&b-4e-8fe-&e-4e-8fa-8.g8e-8c8|1a-
4b-4<c4d8.e-r8.>b-b-b-8b-&:|e-4  "
880 m4="L16c&d&e-&f&d&e-&f&g&e-&f&g&a-&f&g&a-&b-&g&a-&b-&<c&>a
-&b-&<c&d&>b-&<c&d&e-&d&e-&f&g
890 m5="@29gfgf16g16r16g16fgf
900 m6="b-.e-16e-f16e-4b-16b-16b-b-4e-16e-f16e-16&e-4e-f16a-.g
e-c|1a-4b-4<c4d.e-16r8.>b-16b-16b-b-16&b-4e-f16e-4b-16b-16b-b-16
&b-4e-f16e-16&e-4e-f16a-.ge-ce-4fe-16f16&  f8e-2&e-
910 drm1="y3,3y2,10v15@71ev14e"+drm0+"@72e8y2,15@71ee@72e&y2,1
0e@71ee@72y2,10e8y2,15@71ee@72"+drm
920 str  tm[100]="r8o2v13@48p1g-16d16@74p3v14"
930 str tm2[100]="r8o2v13@48p1g-16@74p3v14"
940 str  w1[100]="L16cc8cc8cc8c>gb-<c>L8"
950 str  cs[100]="@34 v12L16o4>b-<c>b-<e-f8gr4"
960 str cs2[100]="o3r8a-ga-ga-gb-g8e-4e-"
970 str tt1[100]="b-4e-f16e-4b-16b-16b-b-4e-16e-f16e-16&e-4e-f
16a-.ge-c
980 /*
990 /*TRK1■□■□■□■□■□■□■□■□■□■□■□■
1000 m( 0)="@v0y48,128 y3,3 v12@88p1"+m1+"18 &e-1&e-2
1010 m( 1)="     y2,24r8.y2,17r16y2,17r16y2,24r16r8116
1020 m( 2)="o4|:3"+drm1+":|
1030 m( 3)="y2,10@71v13ee@72"+drm0+"e8y2,15@71ee@72e&y2,10e@71e
e@72y2,10ey2,15r16y2,18r16y2,10@71v15e&"+drm
1040 m( 4)="|:3"+drm1+":|
1050 m( 5)="y2,10@71v13ee@72"+drm0+"e8y2,15@71ee@72e&y2,10e @71
ey2,15r16y2,10@72e8y2,15@71ey2,10g16 y2,29ry2,29r|:
1060 m( 6)="|:3"+s2+":|y2,10@71ee"+drm0+"eey2,15eeey2,10eeey2,1
0eey2,15rey2,15ry2,15r
1070 m( 7)="|:3"+s2+":|y2,10@71ee"+drm0+"eey2,15eeey2,10ey2,10e
y2,15reey2,15ey2,10e"+drm
1080 m( 8)="|:3"+s2+":|y2,10@71ee@72e8y2,15@71ee@72e&y2,10e@71e
ey2,10@72e8y2,15@71ey2,10v9@88o2b-&y2,15b-&y2,15b-
1090 m( 9)="y2,10b-8g8y2,15g8b-y2,10a-8.&y2,10a-8y2,15o4v13@71e
eee|:2v15"+s2+":|@47o1y2,24c&y2,24c&y2,25c&y2,25cy2,26ry2,26ry2,
27r8y2,10ry2,15r4@71v15y2,10ey2,15ry2,15ro4
1100 m(10)=m(2)+m(3) : m(11)=m(4)+m(5)  :m(12)=m(10)
1110 m(13)="|:"+drm1+":|v15y2,10@71ev14e@72e8y2,15@71ee@72e&y2,
10er8y2,15r8y2,15r8y2,15r16y2,15r16y2,15
1120 m(14)=m(6)
1130 m(15)="|:3"+s2+":|y2,10@71ee"+drm0+"eey2,15eeey2,10ey2,10e
y2,15reey2,15ey2,10ey3,2y2,29@88v10o2b-&y2,29b-y3,3"
1140 m(16)="y2,10b-8y3,2y2,28y3,3g8y2,15g8b-y2,10a-8.&y2,10a-8y
2,15@71v14o4eev10@88o2a-a- y2,10a-8f8y2,15f8a-y2,10g8.&y2,10g8y2
,15@72v14o4ey3,1@71ey3,1y2,29ey2,29ey3,3
1150 m(17)=s2+"y2,10@71ee@72y3,1y2,28y3,3e8y2,15@71ee@72e&y2,10
e@71eey2,10@72e8y2,15@71ey2,10@71ey2,15ey2,15r
1160 m(18)="|:3"+s2+":|@47o1y2,24c&y2,24c&y2,25c&y2,25cy2,26ry2
,26ry2,27r8y2,10ry2,15r4@71v15y2,10ey2,15ry2,15ro4
1170 m(19)=m(10) :m(20)=m(11)
1180 m(21)="|:3"+drm1+":|y2,10@71ee"+drm0+"@72e8y2,15@71ee@72e&
y2,10e@71ee@72y2,10e8y2,15r8y2,15ry2,15r
1190 m(22)="|:3"+drm1+":|y2,10@71ee"+drm0+"@72e8y2,15@71ee@72e&
y2,10e@71ey2,15ey2,10@72e8y2,15ry2,10@72e@71ee
1200 m(23)="|:3"+drm1+":|y2,10@71ee"+drm0+"@72e8y2,15@71ee@72e&
y2,10e@71ee@72y2,10e8y2,15rey2,15ry2,15r
1210 m(24)=drm1+"y2,15ry2,15rry2,10ry2,15ry2,15rry2,10ry2,15rry
2,15rry2,15rry2,15ry2,15r
1220 m(25)="@48o4L16v13|:14r8y3,1y2,28r8r8y3,2y2,29ry2,29r:| y3
,3
1230 m(26)="y2,10r8y2,10ry2,10ry2,14ry2,10rry2,10rry2,14ry2,10r
y2,10ry2,14r8y2,14ry2,14r
1240 m(27)=m(10):m(28)=m(11):m(29)=m(10):m(30)=m(11)
1250 m(31)=m(10):m(32)=m(11):m(33)=m(10)
1260 set(1,33)
1270 /*
1280 /*TRK2□■□■□■□■□■□■□■□■□■□■□■□
1290 m( 0)="@v0y49,120 v12@88p2o2"+m1+"&e-1&e-2 q6 @v0p3
1300 m( 1)="L8 @74v15o3c>b-a-g
1310 m( 2)="|:v15a-4"+tm2+"a-a-16a-"+tm+"v15g4v14"+tm2+"<cc16c"
+tm+"a-4r8.b-b-16b-"+tm+"o3
1320 m( 3)="|1r16c16ccccc16>g16<c16>a.:|<e-e-16e-16e-16e-e-e-16
>b-16<e-16e-16>a-.
1330 m( 4)="|: |:v15a-4"+tm2+"a-a-16a-"+tm+"v15g4"+tm2+"f+f+16f
+8"+tm
1340 m( 5)="v15f4"+tm2+"b-b-16b-"+tm+"|1v15v15g4"+tm2+"<ccc16>g
<c16>g16:|<v15c4"+tm2+"<cc16cr16>a."
1350 m( 6)="|:v15a-4"+tm2+"a-a-16a-"+tm+"v15g4"+tm2+"<cc16c>g<c
16>g16v15a-4"+tm2+"a-a-16a-"+tm
1360 m( 7)="|1<c<c16c16>>b<b16b16>b-<b-16b-16e-<e-16e-16:|o2L16
bbbbbbbbv15b-4v14a4L8
1370 m( 8)="|:v14a-4"+tm2+"a-a-16a- v10@78p1o4e-16d16>b-16g16o2
v15@74g4"+tm2+"<cc16c"+tm+"v15a-4"+tm2+"v14b-b-16b-"+tm+"<"+w1+"
|1a.:|g.
1380 m( 9)="|:v14a-4"+tm2+"a-a-16a- "+tm+"@74g4"+tm2+"<cc16c"+t
m+"v15a-4"+tm2+"v14b-b-16b-"+tm+"|1<"+w1+"a.:|
1390 m(14)="|:o2v14a-4"+tm2+"a-a-16a-v10@78p1o4e-16d16>b-16g16o
2v15@74g4"+tm2+"<cc16c"+tm+"v15a-4"+tm2+"v14b-b-16b-"+tm+"|1<"+w
1+"a.:|<r16ccccL16c>gb-<c>gb-<c
1400 m(15)=" |:L8o2a-4ra-16a-a-16a-4a-16a-16 b-4rb-16b-b-16b-4b
-16b-16 g-4rg-16g-g-16g-4g-16g-16|1a-4r<e-16a-a-e-16a-e-16e-16>:
|r16a-a-16a-a-16a-a-a-16a-16g8.
1410 m(16)="f4r!r2r16g.a-4r1r2r16f.r16b-b-16b-16b-b-b-b-16b-f16
f16<L16ccr>g<ccr>g<cccccL8>g.
1420 m(17)="|:7r1:|r2L8o3c>b-a-gf4"+tm2+"ff16f"+tm+"g4"+tm2+"<c
c16c>g<c16>g16a-4"+tm2+"b-b-16b-"+tm
1430 m(18)="<c16cc16cc16cc16>g16<c16c16>a.v15a-4"+tm2+"a-a-16a-
"+tm+"g4"+tm2+"<cc16c"+tm
1440 m(19)="v15a-4"+tm2+"b-b-16b-"+tm+"<c16cc16dd16e-e-16e-r16>
v15a.
1450 m(20)="v15|:o2a-4"+tm2+"a-a-16a-r8a-v15g4"+tm2+"<cc16c>g<c
16>g16v15a-4"+tm2+"b-b-16b-"+tm
1460 m(21)="|1<c16cc16c>g16<cc16>gr16a.:|<e-16e-e-16dd16cc16<cr
16>>a.
1470 m(22)="v15 L16
1480 m(23)="a-8 a-a- a-8 a-a-& a-a- a-8 a-8 a-a- g8 gg g8 g<c&
cc c8 c8 cc> a-8 a-a- a-8 a-b-& b-b- b-8 b-8 b-8 <c8 cc c8 cc& c
>g< cc c>a& a8"
1490 m(24)="a-8& a-8 r8 r16a-& a-a- a-8 r8 a-8 g8& g8 r8 r16<c&
 cc c8 >g8 <c>g a-8& a-8 r8 r16b-& b-b- b-8 r8 r8 <e-e-& e-e- e-
8 >b-<e-& e-e- e-8 r16>a& a8
1500 for i=4 to 7:m(i+6)=m(i):next
1510 set(2,24)
1520 for ii=0 to 1:FO(2,23):FO(2,24):next
1530 /*
1540 /*TRK3■□■□■□■□■□■□■□■□■□■□■□■
1550 str EGU[256]="@v0@8v8p3b-8<c16e-8.>r8v14@34p1"
1560 m( 0)="@v0 y50,110 L8 @88
1570 m( 1)="r8.v8o2 "+m1+"&e-1&e-2o3@v0
1580 m( 2)="@78v13L16p3v14y50,115b-b-b-8b-&"+m3+"f8.e-&e-2r16<e
-r8r16e8."
1590 m( 3)="L8v12|:r1v12r2@78q5e-d>b-gq7r1|1@v0@84r8v14  >b-<cd
fe-de-@v0:|@v0v14 @84a-2g2
1600 m( 4)="|:r4v14@34o4p1b-4"+EGU+"v13r2L16b-b-b-b-r16L8b-.r4v
14|1b-4"+EGU+"p3
```

```
 1610 m( 5)="r16v13b-16p3v14gv13g-16fv14g-16r16d16e-16f16g16b-8.
:|p1a-4r4r16b-.p3>v13L16b<c+r16d+r16f+r16bv14b-4a4
 1620 m( 6)="q8v10o2@88"+m1
 1630 m( 7)="v14r8.@78o3L16b-b-b-8b-&"+m3+"f4<
 1640 m(11)=m(6)+"&e-2  v14
 1650 m(12)="r4.@33o5L8cc.>b-.<d4d-16c16>b16b-4 f+32 L16g.b-frf8
. r4fg-rb-r<d-rfrg-f4.e-d-rc8d-32c32>b-a-gg-a-8 r8b-4bb-<c8.c8.>
b-<d8. {d-c>b}16   b-4g8b-16f16rf8.
 1660 m(13)="rfrg-b-<d-rf8g-32f8&f32&f8e-8g8a-8b-8<c4d-c>b-8.g&g
8&L32g-&f&e&e-
 1670 m(14)="r2.@88o2p2L16v5b-&v6b-&v7b-&v8b-&v9b-&v10b-&v11b-&v
12b-&v13b-&b-&b- <c&c>b-<c8>
 1680 m(15)="r1r8v10  o2b-8<c>L8b-<c>b-<c>b-.<e-.c.>b-&b-2< c1
 1690 m(16)="v12@88|:3o2"+m1+"&e-2:|
 1700 m(17)="@78 @v80 L8 o4
 1710 m(18)="b-& b-& b-& b-& b-& b-& b-& b-& b-& b-& b-& b- <e-&
 e- d& d >b-& b-& b-& b-& b-& b-& b-& b-& b-& b-& b-& b-& b-& b-
& b-& b-
 1720 m(19)=m(18)
 1730 for i=3 to 5:m(i+5)=m(i):next
 1740 set(3,19)
 1750 for ii=0 to 1:FO(3,18):FO(3,19):next
 1760 /*
 1770 /*ＴＲＫ４□■□■□■□■□■□■□■□■□■□■□■□
 1780 m( 0)="@v0@88 y51,130  v12o2 L8
 1790 m( 1)="|:7r1:|@v0r2v13o4@84"+m4+"v13o4
 1800 m( 2)="@78L16o3r8.v11p3y51,141"+m3+"f8.e-&e-2r16<e-v13c8.L
8r8.v8"
 1810 m( 3)="L8v10|:r1r2q6e-d>b-gq7|1r1@v0@78v10r4r8. ga-b- @v0r
8.@78v10:|@v0r2r4r16v14 @84f2e2
 1820 m( 4)="|:r4v14@34o4p2e-4v13r2r2L16ggggr16L8g.r4v14|1e-4r2p
3>v11e-4d4d-4c4<:|f4r4r16g.v11p3o2b2b-4r4<
 1830 m( 5)="q8v10o2@88"+m2
 1840 m( 6)="v11r8.r8.@78o3L16b-b-b-8b-&"+ m3 +"f4< v8
 1850 m( 9)=m(5)+"&g2 r8.
 1860 m(10)=m(12):m(11)=m(13)+"r2r16 r16
 1870 m(12)="@88o2p1L16v4b-&v5b-&v6b-&v7b-&v8b-&v9b-&v10b-&v11b-
& v12b-&v13b-&v14b- v13<c&v12c>v11b-<c8
 1880 m(13)="v8r1r8"
 1890 m(14)="p3o2b-8<c>b-<c8>b-L8<c>b-.<e-.c.>b-&b-2< c1
 1900 m(15)="&c16& c16 o2@88v12r16r8"+m20+"&g2
 1910 m(16)="|:"+m2+"&g2:|
 1920 for i=3 to 4:m(i+4)=m(i):next
 1930 set(4,16)
 1940 /*
 1950 /*ＴＲＫ５■□■□■□■□■□■□■□■□■□■□■□■
 1960 m( 0)="@v0y52,120
 1970 m( 1)="r1r1r1r1r1r1r1r2 v10@19r32 o4
 1980 m( 2)=m4+"32o5"
 1990 m( 3)="v11L8|:8"+m5+":|
 2000 m( 4)="|:3"+m5+":|gfgf16g16r16g16fga-"+"|:4"+m5+":|"
 2010 m( 5)="@78L1|:o4v10e-&e->b-|1<e-4d4d-4c4:|b2b-4r4
 2020 m( 6)="r4r16"+cs:m(6)="|:"+m(6)+m(6)
 2030 m( 7)=cs2+"|1r2e-4&fc+8.:|
 2040 m( 8)=" @19o4 v14ga-b-<c>a-b-<cd>b-<cde-cde-f
 2050 m( 9)="|:7L8v11"+m5+":|
 2060 m(14)="@19o4v14>b-<cde-cde-fde-fge-ga-<c
 2070 m(15)="@17o4v11|:a-1b-1b-1|1a-1:|a-2.r16g8.v13a-4r2.r2.r16
v11b-8.v13<c4r2.r2.v11r16a-8.
 2080 m(16)=" @78r16v13 o2L16gb-<ce-e-8c v14g8e-8v13 ce-8. <cc>r
g<cc>rgv14<c8c8v13c>g8.>
 2090 m(17)="r8.v8@88"+m1+"&e-2r8"
 2100 m(18)=cs+"r4r16"+cs+cs2+"r2e-4&fc+8.
 2110 m(19)="r4r16"+cs+"r4r16"+cs+cs2+m(8)
 2120 m(20)="@29 @v110 L8
 2130 m(21)="g f g f16g16 r16g16 f g f":m(21)=m(21)+m(21)+m(21)+
m(21)
 2140 m(22)=m(21)
 2150 for i=4 to 7:m(i+6)=m(i) :next
 2160 set(5,22)
 2170 for ii=0 to 1:FO(5,21):FO(5,22):next
 2180 /*
 2190 /*ＴＲＫ６□■□■□■□■□■□■□■□■□■□■□■□
 2200 m( 0)="@v0y53,120  v8r8. @88 o2
 2210 m( 1)=m2+"&g1&g2&g4 r16@v0o5 L8
 2220 m( 2)="@180o3L2v13|:a-1gca-b-|1c.r16a8.:|e-.r16a8.
 2230 m( 3)="v11a-b-gf+fa-gca-b-gf+f1fc4r16a-8. a-1gca-1c4>b4b-4
<c4a-1gcf.g4bb-8r8a4
 2240 m( 4)="|:v13a-1gca-b-<c.> |1r16a8. :|r16g8.
 2250 m( 5)="a-1gca-b-c2.r16a-8.a-1gc  a-b-
 2260 m( 6)=m(3)
 2270 m( 7)="a-1gca-b-<c.>a4a-1gca-b-<c1>
 2280 m( 8)="v12L1a-dg-a-a-dg-a-2.r16a-8.L4 v15fv13r2.r2.r16v13g
8.v15a-r2.r2.v13r16f8.
 2290 m( 9)="L1v14a-c r8.v8@88o2"+m2+"&e-4&e-16
 2300 m(10)="L2@180o3v13a-1gca-b-<c.>r16a8.a-1gca-b-<c.>v14r16a8
.a-1gca-b-<c.>r16a8.a-1gca-b-e-.a4
 2310 m(11)="@180 L8 o3 @v120
 2320 m(12)="a-& a-& a-& a-& a-& a-& a-& a- g& g& g& g c& c& c&
c a-& a-& a-& a- b-& b-& b-& b- <c& c& c& c& c& c> r16a16& a
 2330 m(13)="a-& a-& a-& a-& a-& a-& a-& a- g& g& g& g c& c& c&
c a-& a-& a-& a- b-& b-& b-& b- e-& e-& e-& e-& e-& e- a& a
 2340 set(6,13)
 2350 for ii=0 to 1:FO(6,12):FO(6,13):next
 2360 /*
 2370 /*ＴＲＫ７■□■□■□■□■□■□■□■□■□■□■□■
 2380 m( 0)="@v0y54,130 v12@88o2p1"+m2+"&g1&g1q7 p3
 2390 m( 1)="v12r8L8e-ge-16g16&g4 r4 r8e-ge-16g.b-4. r8e-ge-16g1
6&g4 r4 r8e-<c4>b-4<e-.>g16&
 2400 m( 2)="ge-ge-16g16&g4 r4 r8e-ge-16g.b-4. r8e-ge-16g16g4r4
 e-fga-16b-16&b-4r4
 2410 m( 3)="|:@87q8o5 |:c4cg16f4f16g16f.>b-4b-<f16e-16&e-4cd16e
-16&|1e-de-d16f16&f4d.c>b-.&b-4r2<:|
 2420 m( 4)="e-de-df4d>b-16<a-.g&g2r8g ge-e-f16e-16&e-4r8f16f16f
ddf16e-16&e-4r8g
 2430 m( 5)="ge-e-g16f16&f4r8>b-<<c>b-<c>g16f16e-4 r16g.ge-e-g16
f16&f4r8f16f16fd16db-g16&g4 r8c16d16e-c16e-ce-ce-fg-.e-16g-a-b-4
b-b-b-16&
 2440 m( 6)=m6+"r4"
 2450 m( 7)="@88o2q7"+m(1)
 2460 m( 8)="ge-ge-16g16&g4 r4 r8e-ge-16g.b-4. r8e-ge-16g16g4r4
 2470 m(13)="q8|:8r1:| r8L8c16c16e-c16e-.ce-fg.g.f4e-c16e-.r8ce-
16ce-.ce-c<c.>b-.<c>gfe-cfg16g16&g4r2r2r8. b-16b-16b-b-16&
 2480 m(14)="L8|:"+ttl+"|1a-4b-4<c4d.e-16r.>b-16b-16b-b-16&:| e-
4fe-16f.e-4.r.b-16b-16b-b-16&
 2490 m(15)=ttl+"a-4b-4<c4d.e-16r8.>b-16b-16b-b-16&
 2500 m(16)=ttl+"a-4b-.<c.>gf16e-4&e-16<c.>g.f16e-.&e-2.
 2510 m(17)="@78v9o4b-2<e-4>d4b-1&b-1b-1&b-2<e-4>d4b-1&b-2&b-8.o
3v15L16b-b-b-8b-&
 2520 m(18)="@v127 @78 L8
 2530 m(19)="b-& b- e- f16e-16& e-& e-16b-16 b-16b-16& b-16b-16
b-& b- e- f16e-16& e-& e- e- f16a-16& a- g e- c a-& a- b-& b- <c
& c d >b-16<e-16 >r8 r16b-16 b-16b-16& b-16b-16&
 2540 m(20)="b-& b- e- f16e-16& e-& e-16b-16 b-16b-16& b-16b-16
b-& b- e- f16e-16& e-& e- e- f16a-16& a- g e- c e-& e- f& f16e-1
6& e-& e-& e-& e- r8 r16b-16 b-16b-16& b-16b-16&
 2550 for i=3 to 6:m(i+6)=m(i):next
 2560 set(7,20)
 2570 for ii=0 to 1:FO(7,19):FO(7,20):next
 2580 /*
 2590 /*ＴＲＫ８□■□■□■□■□■□■□■□■□■□■□■□
 2600 m( 0)="@v0y55,110 v12@88o2p2"+m2+"&g1&g1 q7 p3
 2610 m( 1)="v12r8L8ce-c16e-16&e-4 r4 r8ce-c16e-.g4. r8ce-c16e-1
6&e-4 r4 r8 >b-<a-4g4b-.e-16&
 2620 m( 2)="e-ce-c16e-16&e-4 r4 r8ce-c16e-.g4. r8ce-c16e-16&e-4
 r4   >b-<ce-f16g16&g4 r4 r8.v9
 2630 m( 6)=m6+"r16"
 2640 m( 7)="v12@88o2q7"+m(1)
 2650 m( 8)="e-ce-c16e-16&e-4 r4 r8ce-c16e-.g4. r8ce-c16e-16&e-4
 r4 r8.  v9
 2660 m(12)=m(6)+"r8."
 2670 m(16)=ttl+"a-4b-.<c.>gf16e-. v15o3@78L16b-b-b-8b-&"+m3+"f8
 e-2&e-r8.r8.@v120b-b-b-8b-&
 2680 m(17)=" L8
 2690 m(18)=m(19):m(19)=m(20)
 2700 for i=3 to 5:m(6+i)=m(i):next
 2710 set(8,19)
 2720 for ii=0 to 1:FO(8,18):FO(8,19):next
 2730 /*
 2740 m_tempo(116):m_play() /*ﾎﾝﾄﾊ113
 2750 end
 2760 /*----------------------------------------------------------
 2770 func set(trk;int,kazu;int)
 2780   for i=0 to kazu
 2790     m_trk(trk,m(i))
 2800   next
 2810 endfunc
 2820 /*
 2830 func FO(trk;int,no;int)
 2840   str W[256],W2[100]
 2850   int basho,le,p
 2860   W=m(no)
 2870   le=len(W)
 2880   while W<>""
 2890     basho=strcspn(W," "):p=p+basho+1:W2="@v"+itoa(vol(trk)
)+mid$(W,1,basho)
 2900       if vol(trk)<1 then break
 2910       m_trk(trk,W2)
 2920     vol(trk)=vol(trk)-1
 2930     W=mid$(m(no),p,le)
 2940   endwhile
 2950 endfunc
```

## リスト2　ドリカム用コンフィグファイル

```
10=PCMS¥KICK.pcm
14=PCMS¥SD5V15.pcm
15=PCMS¥SD5V13.pcm
17=PCMS¥SD3V15.pcm
18=PCMS¥SD3V13.pcm
24=PCMS¥TOM2_1.pcm
25=PCMS¥TOM2_2.pcm
26=PCMS¥TOM2_3.pcm
27=PCMS¥TOM2_4.pcm
28=PCMS¥ET1V10.pcm
29=PCMS¥ET2V10.pcm
```



## リスト3　SPANISH BLUE

```
10 '           Spanish blue      (DRESS version)
20 '
30 '                  by TM Network
40 '
50 '              Programmed  by  Kuzunari Tanaka
60 '
70 '
80 '
90 '
100 CLS0:INIT:SCREEN:OPTIONSCREEN2:PLAY0:DEFSTR p,d,o,n,w,t,c,a,
g
110 DIM p(40)  :GOTO160
120 '
130 LABEL"!"
```

```
140 READ z:IF z<>255 THEN PLAY p(z);:GOTO130
150 RETURN
160 '
170 PLAY"t165 ";
180 '
190 '   Drum
200 '
210 d1="i1o1ci3c
220 '
230 op="I6V16O4L8 rcc4c4cc  I3o1 r4.cr4cc
240 p(0)="p1k3"+op
250 p(1)="14o1v16p3 "+d1+d1+d1+"i5r8i1c8i3c":p(1)=p(1)+p(1)+p(1)
260 p(2)=d1+"i1ci3c8c8
270 p(3)=d1+d1+d1+"8i1ci1c8i3c
280 p(4)=p(3)+"8c8
290 p(5)="i1c8c8r8c8r2
300 p(6)=d1+"8i1c8ri3c
310 p(7)="i1c8c8i3c8i1c8ri3c"
320 p(8)="i1c8c8i3c8i1c8r8c8i3c
330 p(9)=p(8)+"8c8
340 p(10)=d1+d1
350 p(11)=d1+"8i1c8r8c8i3c
360 p(12)=d1+"8i1c8i3c8c8c8c8
370 p(13)=p(11)+"8c8
380 p(14)=d1+"8c8c8c8c8c8
390 p(15)="r2.i3c
400 p(16)="i1c8c8i3c"+d1
410 p(17)="i1c1
420 '
430 RESTORE 440:"!"
440 DATA 0,1,2,1,2,1,2, 3,4
450 DATA 5,6,6,7,6,7,6, 8,6,7,6, 8,6,7,6, 9                              '
x1
460 DATA 10,11,10,11, 10,11,10,11, 10,11,10,11, 10,11,10, 12
470 DATA 10,11,10,11, 10,11,10,13, 10,11,10,11, 10,11,10, 13, 10
,11
480 '
490 DATA 5,6,6,7,6,7,6, 8,6,7,6, 8,6,7,6, 9                              '
x2
500 DATA 10,11,10,11, 10,11,10,11, 10,11,10,11, 10,11,10, 12
510 DATA 10,11,10,11, 10,11,10,13, 10,11,10,11, 10,11,10, 14
520 '
530 DATA 10,11,10,11, 10,11,10,11, 10,11,10,11, 10,11,10,11
540 DATA 10,11,10,11, 10,11,         15                                  'd
.s.
550 DATA 10,11,10,11, 10,11,10,13, 10,11,10,11, 10,11,10,13     '
to coda
560 DATA 16,11,10,11, 10,11,10,11, 16,11,10,11, 10,11,10,11
570 DATA 16,11,10,11, 10,11,10,11, 16,11,10,11, 10,11,10,11,  17
 ,255
580 '
590 PLAY":";
600 '
610 ' Bass
620 '
630 p(0)="p2k7"+op                  'clap ホシ゛ョ
640 p(1)="i9o218v13p3 "+STRING$(7,"r1")
650 p(2)="e-r4.r2 gr4.r2 e-r4.r4 f>f<g>g<r2. e-r4.r4 f>f<g>g<r4.
f4.e-r4.r4f>f<
660 p(3)="g2g2 g4.grgr4 ":p(3)=p(3)+p(3)
670 p(4)="r4ggr2 g4ggr2 d4>d<dr2 dd>d<dr2 f4.fr2 ff>f<fr2 e-4.e-
r2
680 p(5)="e-e->e-<e-re-e-4 g4.gr2 gg>g<gr2 d4.dr2
690 p(6)="dd>d<drdd4 f4.fr2 ff>f<fr2 e-4.e-r2
700 p(7)="dd>d<drdd4 f4.fr2 ff>f<fr2 c4cde-r4.
710 p(8)="e-e->e-<e-r2 f2f2 e-4.e-re-e-4 d2d2
720 p(9)="d4.drd4. e-2e-2 d4.b-rfd4 c2c2 c4r4rc4.
730 p(10)="f2f4>f4< e-4.e-re-4. d2d2 d4>d<drdd4
740 p(11)="e-2e-2 d4.drdd4 c2c2  f4.ff+f+f+4
750 p(12)="e-2e-2 d4.drdd4 c2c2  rffff+f+f+f+
760 p(13)="e-4.e-f4.f g4.frfr4   e-2f2 ggggrfrf
770 p(14)="e-4.e-f4.f g4.frf>f4< e-2f2 gfgab-agf
780 p(15)="e-4>e-<e-f2 g4.frfr4 e-4>e-<e-f4f4 ggfgggfg
790 p(16)="e-4>e-<e-f4.f g4.frf4. e-4>e-<e- f2 gfgab-agf g4r4g4r
4 g4ggrgg4
800 p(17)="e-4>e-<e-f4.f g4.frf4. e-4>e-<e- f2 ggggggff
810 p(18)="g4g4g4g4 g4.grg4.
820 p(19)="g2g2 g4.grg4.
830 p(20)="g4g4g4g4 f+4.f+rf+r4 r1
840 p(21)="e-4>e-<e-f4.f g4.frf4. e-4>e-<e- f2 gggg>dfgf<
850 p(22)="e->e-4<e-f4.f g4.grg4f e-4>e-<e-f2 g4.grg4f
860 p(23)="e-4.e-f4.f g4.frfrf e-4>e-<e-f2 gggab-agf
870 p(24)="e-4>e-4<f4>f<f g4.a4b->c<b- e-4>e-<e-f4>f<f ga4b-4agf
880 p(25)="e-4>e-<e-f4>f<f g4>gfrffe-< e->e-<e-4f4.f gg4.ab-ag
890 p(26)=STRING$(8,"r1")
900 p(27)="i2o5v13 k7 c1
910 '
920 RESTORE 930:"!"
930 DATA   0,1,2,2, 3
940 DATA  4,5,6,8,9,10,11,13,14,15,16              'x1
950 DATA  4,5,6,8,9,10,12,13,14,15,17              'x2
960 DATA  18,19,19,19,18,19,19,19, 19,19,20       'd.s.
970 DATA  13,14,15,21                              ' to coda
980 DATA  22,23,22,23,24,25,26,27,255
990 '
1000 PLAY":";
1010 '
1020 '  Bass  2
1030 '
1040 p(0)="p3k10"+op               'clap ホシ゛ョ
1050 p(1)="i9o118v14p3 "+STRING$(7,"r1")
1060 p(27)="r1
1070 '
1080 RESTORE 930:"!"
1090 '
1100 PLAY":";
1110 '
1120 '   Main  1
1130 '
1140 p(0)="r1r1
1150 p(1)="i9o118v12p3k2 "+STRING$(7,"r1")
1160 p(4)="i8o418k3v14 rb-b-ab-4r4 rgab->c<b-ab- a2.fd r2rddd
1170 p(5)="a4.b-a4r4 rab->cdc<b-a g2rg>dc< rb-rarb-ag
1180 p(6)="aaab-4.>c4< aaab-4.>c4 cccd4.f4< b-4r4.>ddd
1190 p(7)="ddde-4.f4 ddde-4.f4 dcdc&c2& c4r4r<b-b-b-
1200 p(8)="ddde-4.f4 ddde-4e-f4 dcdc&c2& c4r4r<b-b-b-
1210 p(9)="aaab-4.>c4< aaab-4.>c4 cccd4df4< b-4r4.>ddd
1220 p(10)="ddde-4.f4 ddde-4.f4 dcdc&c2& c2r4r f
1230 p(11)="gr4gfr4d de-dc<b-r4>f gr4gfr4d de-dc<b->r4d
1240 p(12)="e-r4cdr4d de-dc<b-rb-a g4>g4f4e-4 d<ggg4.r>f
1250 p(13)="e-r4cdr4d de-dc<b-rb-a g4>g4f4e-4 d<g4g&g2
1260 p(14)=p(13)+"r1r1"
1270 p(15)=STRING$(22,"r1")
1280 p(16)="r2. i8o4v14q8 r>f
1290 p(17)="i8o5v14q8 gr4gfr4. i6v16o4cc4c4cc i8o5v14q8 f
1300 p(18)="gr4gfr4. re-re-re-rf
1310 p(19)="gr4gfr4. r2.r4
1320 p(20)="r1r2.rde-r4e-dr4.r1
1330 p(21)="r1r1r1r1
1340 p(22)="r1
1350 p(23)="i16o7v12s4,1,0,8=3k7 c2.r4
1360 p(24)="i16o6v12s3,1,0,1=3q1 c8c8c8c8c8c8c8   =0q8 r8
1370 p(27)="i12o4v12e-16f16f16g16g16a16b-16>c16 d16e-16f16f16 g8
r8
1380 p(28)="i12o3v12s4,5,0,8=3c16c8.&c8c16c16&c4 c16c8.&c8c16c16
&c4c4=0 r4
1390 '
1400 RESTORE 1410:"!"
1410 DATA   0,1,2,2,3                                            'bas
s ホシ゛ョ
1420 DATA   4,5,4,5,6,7,9,10,11,12,11,14                        'x1
1430 DATA   4,5,4,5,6,8,6,10,11,12,11,13                        'x2
1440 DATA   21,21 ,22,23 ,22,23 ,22,23 ,22,24
1450 DATA   22,27,28,22,22,16                                    'd.s
.
1460 DATA   11,12,11,12                                          'to
coda
1470 DATA   17,17,17,18,17,17,17,19,20,21,21,21,22
1480 DATA  255
1490 '
1500 PLAY":";
1510 '
1520 '  main 2
1530 '
1540 p(2)=STRING$(7,"r1")
1550 p(3)="r1r1r1r1
1560 p(25)="r8.
1570 p(4)="i8o418k7 v7    rb-b-ab-4r4 rgab->c<b-ab- a2.fd r2rddd
1580 p(10)="ddde-4.f4 ddde-4.f4 dcdc&c2& c2r8.  v14  d
1590 p(11)="e-r4e-dr4<b- b->c<b-agr4>d e-r4e-dr4<b- b->c<b-agr4
r8.v7 >d
1600 p(12)="e-r4cdr4d de-dc<b-rb-a g4>g4f4e-4 d<ggg4&g16 v14 >d
1610 p(16)="r2 r8  i8o4q8 v14 >d
1620 p(17)="i8o5v14q8 e-r4e-dr4.  i6o4v16 cc4c4cc    i8o5q8v14 d
1630 p(18)="e-r4e-dr4. re-re-re-rd
1640 p(19)="e-r4e-dr4. r2.r4 r8. v7
1650 p(23)="i16o7v00s4,1,0,8=3k3 c2.r4
1660 p(30)="r1r1r1r2.r8
1670 RESTORE 1680:"!"
1680 DATA   0,1,2,2,3                                            'bas
s ホシ゛ョ
1690 DATA   25,4,5,4,5,6,7,9,10,11,12,11,14                     'x1
1700 DATA      4,5,4,5,6,8,6,10,11,12,11,13                     'x2
1710 DATA   30,21,22,23,22,23,22,23,22,24
1720 DATA   22,27,28,22,22,25,16                                 'd.s
.
1730 DATA   11,12,11,12                                          'to
coda
1740 DATA   17,17,17,18,17,17,17,19,20,21,21,21,22
1750 DATA  255
1760 '
1770 PLAY":";
1780 '
1790 '  synth 1
1800 '
1810 p(0)="r1 k7 p1
1820 p(1)="r1r1r1r1r1r1r1 q4 s2,3,0,5 =2 h3 v7
1830 p(2)="i11o614v8 <b->gfd8c8 <b-8>gf4.d8c8
1840 p(3)="<b->gf4c
1850 p(4)="q6 g>e-d2 <g8>e-d8&d2<
1860 p(5)="i13o4v614=0q7 d8f8b-8>d8&d2&d1<
1870 p(6)="c8f8a8>c8&c2&c1<
1880 p(7)="i15v13o5 b-4.a4.f4 d4.c4.<b-4> f4.g8&g2 r1r1r1 v6
1890 p(8)="<b-4>g4f4d8c8<b-8>g4f8&f2
1900 p(9)=STRING$(8,"r1")
1910 p(10)="
1920 RESTORE 1930:"!"
1930 DATA   0,0,1,1,2,2,2,3,4,4
1940 DATA   5,5,6,6,5,5,6,6,0,0,7,0,0,7,2,2,2,2,2,2,2,2,4
 'x1
1950 DATA   5,5,6,6,5,5,6,6,0,0,7,0,0,7,2,2,2,2,2,2,2,8
 'x2
1960 DATA   9,9,1
 'd.s.
1970 DATA   2,2,2,2,2,2,2,2
 'to coda
1980 DATA   2,2,2,2,2,2,2,2,8,8,0
1990 DATA  255
2000 '
2010 PLAY":";
2020 '
2030 '  synth 2
2040 '
2050 '
2060 p(0)="r1 k3 p2
2070 p(10)="r96
2080 '
2090 RESTORE 1930:"!"
2170 '
2180 PLAY":";
2190 '
2200 '  percussion
```

```
2210 '
2220 wb="i4o7p2c4
2230 no="i5o7p1c8c8
2240 cb="i7o4p1c8c8
2250 tb="i2o8p2 v8 c8 v11
2260 '
2270 p(0)="p314v11  r1r1
2280 p(1)="r2"+wb+no+cb+"r4 r8"+tb+wb
2290 p(2)="r2"+wb+no
2300 p(3)="r2 r8i4o7p2c8c4 r2"+wb+no+no+"r2. r2"+wb+no
2310 p(4)=cb+"r4."+tb+tb+"r8 r2"+wb+no
2320 p(5)="r1
2330 '
2340 RESTORE 2350:"!"
2350 DATA  0,1,1,1,2,1,1,1,2,1,1,1,2,3     'Intro
2360 DATA  4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4   '
x1
2370 DATA  4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4   '
x2
2380 DATA  4,4,4,4,4,4,4,4 ,4,4,4,5                          '
d.s.
2390 DATA  4,4,4,4,4,4,4,4 ,4,4,4,4,4,4,4,4
2400 DATA  4,4,4,4,4,4,4,4,5
2410 DATA  255
2420 '
2430 PLAY":";
2440 '
2450 ' Hi-hat   (PSG 1)
2460 '
2470 p(0)="y7,56y7,177y6,3o5v12 s4,1,15,0 =3   r1r1
2480 p(1)="14 cccc
2490 p(2)="14 ccrr
2500 p(3)="18 cccccccc
2510 p(4)="18 ccccrrrr
2520 p(5)="o7=2 c1
2530 p(6)="r1
2540 RESTORE 2550:"!"
2550 DATA 0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
'Intro
2560 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,2
2570 DATA 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1,1
' x1
2580 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
1,1,1,1,2
2590 DATA 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,4
' x2
2600 DATA 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,6
' d.s.
2610 DATA 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3
' to coda
2620 DATA 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,
3,3,3,3,3,3
2630 DATA 5,255
2640 '
2650 PLAY""
2920 '
2930 '
2940 '
```

## （善）のゲームミュージックでバビンチョ

ミステリーサークルが宇宙人のメッセージかどうかはさておき，とにかく秋であります。夏のあいだはほとんど毎日のようにやっていた怪奇特集やUFO特集や稲川なんとかの顔も，最近はお目に掛かれなくなりました。あぁ，もう秋なんですね。それはさておき，やい，池田貴族とやら，おまえが紹介したミステリースポットは，ソフトバンクのあるビルの真横だぞ。

**●地層階級王国～まほろばの壁を越えて～**
**CD：PCCG-00140　ポニーキャニオン**
**2,800円(税込)**

本来ならこういう類のは守備範囲外なんだけれど，ほかのGMのCDと一緒に送られてきたので一応紹介しよう。

これは「ゲームCD」というもので，まあ，いってみればゲームブックのCD版といった感じ。聴取者が主人公に扮し，各トラックごとに収められた問いかけに答えていきながら，お話の結末を目指すわけ。たとえば「水がある。飲むならトラック5へ，飲まないならトラック10へ」といった内容が延々と収録されているのだ。

ゲームブックと違う点といえば(CDですから)，効果音やBGMがリアルなのと登場人物が声優さんたちの声で喋りまくるところかな。で，ポニーでは今後もこのゲームCDをリリースしていくそうなので，ゲームブックで物足りなかった人，パソコンを持っていないけどアドベンチャーゲームをやりたい人などにお勧めだ。

お勧め度　？

**●交響組曲アクトレイザー/古代祐三**
**CD：ALCA-182　アルファレコード　2,500円(税込)**

エニックスより発売中のスーパーファミコン用ゲーム「アクトレイザー」のBGMをオーケストラで演奏したもの。原曲はスーパーファミコンの音源を極限まで駆使したと思われる素晴らしいものであったが，やはりストリングスなどの持続系の楽器音は苦しいものがあった。

ま，そういうわけで，きっと今回のアルバムが，作曲者のイメージした本来のサウンドなのではないだろうか。聴いた感じでは，曲の構成の大幅な変更はないようで，純粋に原曲を生楽器に置き換えたといった感じがする。ここらへんは賛否両論かもしれないが，私は原曲が一層深まった感じを受けた。

・同時発売に「ベアナックル/古代祐三」もあるぞ。

お勧め度　8

＊

UFO特集で思い出したが，ナムコの新作「スターブレード」は凄いね。私はかねてから，宇宙3Dものは巨大宇宙戦艦の下をビューと飛べるものじゃなけりゃねぇ，と他人からは理解されがたい意味不明な願望があったのだった。それらの願望をこの「スターブレード」はいとも簡単にみんな叶えちゃってくれてるので，私は初め小便をちびりそうになった。制作者は宇宙ものに対する考えが，かなり私と似ているんじゃないかと思うぞ，いやぁ，まいった。ボードのCPUは68030×数個だそうだけど，いつかX68000の後継機種でもこんなゲームが走るといいな。さあ，みんなもゲームセンターに行ってさっそく小便をちびろうぜ。いえい，ゴーゴー。

## GAME-MUSIC FESTIVAL '91に行ってきたぞ！っと

もう街はすっかり秋。まだなんとなく暑いような気もするけど，でも秋です。深く考えないように。

さて，7月30・31日，中野サンプラザにおいて「GAME-MUSIC FESTIVAL '91」が行われました。こーゆーのってめったにナマじゃやらないっていうし，しかもあたしは初体験。不安と期待を胸に会場へ向かったのでした(31日だけだけど)。

会場に着くと，あらあらもう「矩形波倶楽部」の演奏が始まってる……。このバンドの音はホントにフュージョンしてる。自然に耳に入り込んできてソツなくこなして終わるってカンジ。しゃべりはイマイチだったけど落ち着いて聴けたので，遅刻組のあたしはホッとしてしまった。

2番手は「ALFH LYRA」。いやー，このバンドは若いせいか元気元気。テクは普通なんだけど，いけいけやっちゃえ～っ，の勢いで押しまくる。観客もコレもんで拳を振り上げる。いいなあ，若いって。曲は「戦場の狼II」「ストII」など。初めてのライブということで，ちょっとあがってるのかな，と思うふしは多々あったけど，いいぞ，おねーさんはこーゆーの好きだぞ。がんばれ。

トリはやっぱりというか当然というか「S.S.T. BAND」。観客のほうも待ってました！　とばかり全員総立ちのノリノリ。聴かせるノらせるMCバッチリと三拍子揃ってるもんね。さ～すが，普通の会社員にしとくのもったいないわ。「AFTER BURNER」ほか数曲演奏し「LIKE THE WIND」でシメ。もちろんアンコールもやっちゃってくれました。

しかしラストのジョイントには笑った。「HIGH WAY STAR」ですか。みんなロック少年だったのねぇ。J.D.K.BANDの岸本さんのヴォーカルって初めて聴いたわ。うん，容姿もだけどヘビメタで気に入った。こんど取材しようっと！　（出口香）

ALFH LYRA

S.S.T. BAND

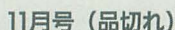
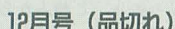
ジョイント

▶「Advanced Magical World」のCGを見て，ますますCGに興味がわきました。私もやってみたいです。　中村　圭介(16)神奈川県

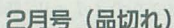
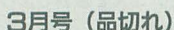
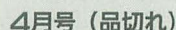
# BACK ISSUES バックナンバー案内

ここには1990年10月号から1991年9月号までをご紹介しました。現在1990年10，1991年1，5～9月号の在庫がございます。バックナンバーおよび定期購読の申し込み方法については，172ページを参照してください。

1990

## 10月号
特集　電子音楽術入門
連載　ショートプロぱーてぃ/Z80's Bar/DōGA・CGA
マシン語プログラミング/ハードウェア工作入門
清水和人流プログラミング道場
●荻窪圭の大人のためのX68000
●中森章のようこそここへC言語
LIVE in '90　Rise And Fall/PARADOX/キューピー3分クッキング
THE SOFTOUCH　ワールドコート/ルーンワース/闇の血族/提督の決断
全機種共通システム　ライブラリアンWLB

## 11月号（品切れ）
特集　理科系のGAME REVIEW
連載　Z80's Bar/DōGA・CGA/カードゲーム
マシン語プログラミング/ハードウェア工作入門
PurePASCAL/X-BASIC調理実習
ようこそここへC言語/INTEGRAL XI
●荻窪圭の大人のためのX68000
LIVE in '90　ピラミッドソーサリアン/ザ・スキーム
THE SOFTOUCH SPECIAL　ラグーン/幻獣鬼/サイバリオン/GUNSHIP他
全機種共通システム　スクリーンエディタEDC-T

## 12月号（品切れ）
特集　XCのための傾向と対策
連載　X-BASICプログラミング調理実習/ハードウェア工作入門
マシン語プログラミング/ショートプロぱーてぃ/Z80's Bar
大人のためのX68000/ようこそここへC言語/INTEGRAL XI
●シミュレーションプログラミング入門
●特別企画アナログジョイスティックの制作
LIVE in '90　グラディウスIII/メタルサイト
THE SOFTOUCH SPECIAL　イメージファイト/ジェミニウイング/NAIOUS他
全機種共通システム　STACKコンパイラ

1991

## 1月号
特集　急接近! SX-WINDOW
特別付録　謹賀新年PRO-68K（5″2HD）
連載　ハードウェア工作入門/シミュレーションプログラミング入門
DōGA・CGA/ショートプロぱーてぃ/大人のためのX68000
PurePASCAL/清水和人流プログラミング道場/X-BASIC調理実習
LIVE in '91　めぞん一刻/涙で綴るパパへの手紙
THE SOFTOUCH　ソル・フィース/銀英伝II/続ダンジョン・マスター他
製品紹介　光磁気ディスクCZ-6MOI
全機種共通システム　ブロックアクションゲームCOLUMNS

## 2月号（品切れ）
特集1　グラフィックの〝実験的〟手法
特集2　SX-WINDOWプログラミング
連載　ハードウェア工作入門/シミュレーションプログラミング入門
マシン語プログラミング/大人のためのX68000/Z80's Bar
ショートプロぱーてぃ/INTEGRAL XI/ようこそここへC言語
●1990年度 GAME OF THE YEARノミネート発表
LIVE in '91　Misty Blue/スプーンおばさん
THE SOFTOUCH　栄冠は君に/KLAX/ダイナマイト・デューク他
全機種共通システム　ダイスゲームKISMET

## 3月号（品切れ）
特集　MIDI & MUSIC PROCESSING
連載　ハードウェア工作入門/シミュレーションプログラミング入門
マシン語プログラミング/大人のためのX68000/Z80's Bar
ショートプロぱーてぃ/DōGA・CGA/C言語/PurePASCAL
●SXLIFE完結編/ウィンドウシステム大比較
●周辺機器新製品紹介
LIVE in '91　戦いの兜/LITTLE WING/リゾ・ラバ/花
THE SOFTOUCH　アトミック・ロボキッド/スペースローグ他
全機種共通システム　アクションゲームMUD BALLIN'

## 4月号（品切れ）
特集　人とゲームのインタフェイス
連載　DōGA・CGA/シミュレーションプログラミング入門
ハードウェア工作入門/ようこそここへC言語/Z80's Bar
ショートプロぱーてぃ/清水和人流プログラミング道場
●新連載　吾輩はX68000である/よいこのSX-WINDOW講座
●決定!　1990年度GAME OF THE YEAR
LIVE in '91　Easy Come, Easy Go!/シシリエンヌ
THE SOFTOUCH　メルヘンメイズ/中華大仙/スライス他
全機種共通システム　SLANG用カードゲームDOBON

## 5月号
特集　新登場!　X68000XVI/XVI-HD
特別付録　黄金週間PRO-68K（5″2HD）
第6回　言わせてくれなくちゃだワ
連載　ハードウェア工作/ようこそここへC言語
大人のためのX68000/X68000マシン語プログラミング
ショートプロぱーてぃ/マシン語カクテル in Z80's Bar
LIVE in '91　ブービーキッズ/NO.NEW YORK
THE SOFTOUCH　マーブル・マッドネス/シグナトリー/石道他
全機種共通システム　実数型コンパイラ言語REAL

## 6月号
特集　初心者のための環境構成術
創刊9周年記念Oh!Xアンケート結果大分析大会その1
連載　ハード工作/大人のためのX68000/Z80's Bar/DōGA
ようこそC言語/ショートプロぱーてぃ/SX-WINDOW
吾輩はX68000である/マシン語プログラミング
●響子 in CGわ～るど
LIVE in '91　暴れん坊将軍/ナディア/POWER HALL他
THE SOFTOUCH　パロディウスだ!/遥かなるオーガスタ/ノスタルジア他
全機種共通システム　S-OS 6周年記念　Small-C　処理系の移植

## 7月号
特集　Personal Tool,BASIC
別冊付録　X-BASIC　ポケットリファレンスブック
連載　大人のためのX68000/ハード工作/響子 in CGわ～るど
ショートプロぱーてぃ/SX-WINDOW/吾輩はX68000である
ようこそC言語/Z80's Bar/マシン語プログラミング
●X1用ゲーム　The Master of Payment
LIVE in '91　今すぐKISS ME/歩いていこう
THE SOFTOUCH　パロディウスだ!/ファランクス/スコルピウス/AIII他
全機種共通システム　実数型コンパイラ言語REAL ソースリスト編

## 8月号
特集　印刷の世界へ
連載　大人のためのX68000/SX-WINDOW/ようこそC言語
響子 in CGわ～るど/ハード工作/ショートプロぱーてぃ
吾輩はX68000である/マシン語プログラミング
●X68000カードゲーム　七並べ
●X1用ゲーム　DEFEAT2
LIVE in '91　パワードリフト/イースIII/TURBO OUTRUN
THE SOFTOUCH　黄金の羅針盤/サイレントメビウス/パロディウスだ!他
全機種共通システム　Small-C ライブラリの移植

## 9月号
特集　Brush up your MAGIC.
連載　マシン語プログラミング/DōGA/Z80'sBar/ショートプロ
響子 in CGわ～るど/ハード工作/シミュレーション入門
吾輩はX68000である/大人のためのX68000/C言語
●X1用ゲーム Manual Runner
●ANOTHER CG WORLD
LIVE in '91　One/WHITE MANE
THE SOFTOUCH　イース/生中継68/アークス・オデッセイ他
全機種共通システム　SLANG用NEWファイル入出力ライブラリ

〈対応機種一覧〉●MZ-80K/C/700/1500●MZ-80B/2000●MZ-2500/2861●X1●X1turbo/Z●PC-8001/8801/88●SMC-777/C●PASOPIA/5●PASOPIA 7●FM-7/77/AV●PC-286/386/9801/98●X68000
掲載されたプログラムの利用には各機種用のS-OS"SWORD"システムが必要です。

# 第111部 Small-C活用講座（初級編）

**●Small-Cアフターケアのお知らせ**

S-OSで初めてのCコンパイラであるSmall-Cに，数々の反響のお便りをいただいています。さっそく使っているというお便りに，時間をかけて用意しただけのことはあったとスタッフ一同胸を撫でおろしているところです。なかでもバグ情報を寄せてくださった皆さんと，そのバグを取り除いてくださった皆さんには本当に感謝しています。#includeで取り込むファイルを< >で囲むとエラーとなる，コンパイラの出力するアセンブラファイルである〜.ASMの中にDS疑似命令が入っているとアセンブルできないなど，いくつかの情報が寄せられています。これらのバグについては，今月のバグ出しをご参照ください。

また，SOROBANやMAGIC,TURTLE用のライブラリはどうだろうか，という提案も届いており心強いかぎりです。C言語は言語仕様が小さい分，ライブラリの充実が不可欠です。どんどんご投稿ください。ライブラリのアイデア，Small-C用のアプリケーションプログラムともども皆さんのお便りをお待ちしています。

**●Small-C入門講座**

ライブラリの整備によってC言語が本格的に始動しました。編集部にも「これを機会にC言語を本格的に勉強してみたい」というお便りが届いています。Z80というCPUの性格と64Kバイトというメインメモリの制約上，フルセットのCコンパイラというのは少々つらいものがあります。8ビットパソコン用のCコンパイラの多くがSmall-Cと同様にサブセット版となっているのはそのためです。だからといって，サブセットでは実用にならないなんてことはまったくありません。実際8ビットマシンのCコンパイラを使って開発された市販ゲーム(それもかなり有名な)も存在するくらいです。考えようによっては，余計な枝葉がない分だけ，Small-Cは初心者にとってくみしやすい相手といえるのではないでしょうか。

Small-CでC言語の入門をしたいという皆さんのために，Small-C活用講座を用意することにしました。初回の今回は，簡単なプログラムを例にC言語の世界に触れてみていただきたいと思います。中森氏の「ようこそここへC言語」はX68000用のCコンパイラを対象に書かれていますが，基本はX68000用だろうとSmall-Cだろうと変わりません。併せて取り組んでみてください。次回の活用編では，Small-Cならではといった使い方や，S-OSをSmall-Cから直接操作する方法といったものを紹介します。いまやS-OSの標準言語となった観のあるSLANG,REALともども，皆さんと一緒に大きく育てていきたいものです。

**●S-OSの系譜(25)**

1987年12月号は，正真正銘のOh!CZスペシャルでした。Oh!MZ時代にXシリーズの情報を盛り込んだOh!CZの企画が実行されたことがありましたが，ついに，この12月号より誌名がOh!Xに変更されたのです。

Oh!Xになったとはいえ，THE SENTINELは相変わらずS-OSシステムの見張り番を続けます。12月号では，どんどん増殖の手を伸ばしていたS-OS"SWORD"が，ついに東芝の8ビットマシンPASOPIA7をその傘下に収めました。これをもってとどまるところを知らないかのように見えた一連のS-OS移植攻勢はひとまず終わりました。国内の主要パソコンメーカーの8ビットマシンを網羅するS-OSの拡がりは，名実ともに実働する8ビットマシンのOSとして最大のユーザー規模を誇ることになったのです。

同時にこの12月号では，MAGICを使ったちょっと面白いプログラムが掲載されました。MAGICは単独でも動作しますが，より容易に扱えるようS-OS上の各種の言語がこれをサポートしています。

ところがシステム開発にのめり込む人の性なのか，これらの言語はことごとく再帰ができるようになっています。「それならタートルグラフィックができたほうが面白いのではないか」と考えた読者がいたのです。タートルグラフィックパッケージTURTLEはこうして誕生しました。このパッケージはMAGICを利用したもので，MAGIC同様機種を問わず使用することが可能でした。magiFORTH用のワードも発表され，FORTHユーザーには2度おいしいパッケージでした。

12月号にはアフターケアとして，turbo"SWORD"用のラインプリントルーチンが掲載されています。これはS-OS初のスクリーンエディタE-MATEが画面表示の高速化を図って採用した，文字列を直接VRAMに書き込むルーチンです。この結果，X1turboでも快適なエディタ環境が用意されることとなりました。

THE SENTINEL

全機種共通
S-OS"SWORD"要

# Small-C
## 活用講座（初級編）

Ishigami Tatuya
石上 達也

**今月号と来月号にわたってSmall-Cを移植した石上氏がC言語の解説をしていきます。今月はサンプルプログラムを通して一般的な解説をして，Small-Cの具体的な活用を来月号で行う予定です。**

先日，編集部での会話。
担当「8月号にのっけるライブラリだけど，ファイル名の大文字，小文字って関係あるの？」
私　「ええ，ディスクに入ってたとおりにしてください」
担当「どうしても？　いまから全部大文字に直せない？」
私　「リンカは，ちゃんと大文字小文字区別しますからね。なんでです？」
担当「MZ-80が……」
私　「……」

というわけでごめんなさい。Small-CはMZ-80K/C系で使用することができません。上で述べたように大文字小文字の問題もありますが，それ以前にメモリが圧倒的に足りないので素直にあきらめてください。MZ-80K/C系以外でも_MEMAXが$F000付近までない機種はかなり危ないと思っていいでしょう（X68000と比べると，なんてのどかな世界なんだ）。

## C言語について

いまさらOh！Xの読者に向かってC言語とは何か，でもないでしょうが日頃使っていて他の言語と違うな，と思った点を挙げてみます。

1)　比較的文法がきれいである。
2)　言語仕様が小さい（その分，ライブラリを充実させる必要がある）。
3)　演算子が豊富である。
4)　ポインタが容易に使える。
5)　複雑なデータ構造が扱える。

1)によってコンパイラは，非常にコンパクトに設計できます。BASICでSINの値を求める，

sin(a)

という関数があるとします。ここで問題になるのは変数aの値です。ところが，

varptr(a)

の場合には，変数aの中身ではなく変数のアドレスが問題となります。このようにBASICでは，同じような文法規則が適用されても，その内容によって意味が全然違ってきてしまいます。その最たるものが，FOR文で，

for i=0 to 10 step 9

の場合，まず予約語「for」を拾ったら，次にくるのは式ではなく変数（名）でなければなりません。そして，そのあとには必ず（あまり意味がないと思われるが）「=」があるかどうかチェックして……というように，まったく独自の文法規則を作っています。これに対してC言語では，

for(式1；式2；式3)　文

というように，非常にきれいに定義されています。

2)についても同様です。BASICなど入出力関係はステートメントとして，あらかじめ言語仕様に用意しておくという手段を用いて行っていました。これは，1)とダブりますが，結構，コンパイラを大きくしてしまうのです。その結果，コンパイラ自身の移植性を著しく低下させてしまいます。

C言語では，このような組み込みステートメントを提供するのではなく，各自が用意するようにして，コンパイラの肥大化を回避しているのです。各自が用意するといっても，たいていはコンパイラを作った人がライブラリとして作っておいてくれますから，プログラマはまったく意識する必要はありません。また，その仕様が気に入らなければ，自分で作り直してしまうことも普段プログラミングをするような感覚で可能なのです。

## 8ビット機への移植について

本来，C言語というのはDEC社のPDP-11というミニコンで動くように設計されたものです。PDP-11とSWORDが想定しているCPUのZ80では，構造がまったく違います。私と島田紳助ぐらい違うのです（全然説明になっていないって）。Z80というのはC言語を走らせる環境として，かなり厳しい部類に入ります。このような環境でC言語を走らせるためには，いくつかの機能を省略させなくてはなりません。Small-Cが省略したのは，

構造体，共用体
実数の扱い
ビットフィールド

などです。構造体，共用体は配列をうまく使うことで，切り抜けてください（実例は11月号で説明します）。実数の扱いについては，実数演算パッケージを作ることで解決できるでしょう。ビットフィールドは，

ANDやORなどをうまく使うようにすればなんとかなります。

## サンプルプログラム1

ここでは，C言語の解説ではなく，サンプルプログラムを使って，C言語のノリを知ってもらおうと思います。C言語について詳しく知りたい方は，中森氏の連載「ようこそここへC言語」や，泉大介氏の連載「吾輩はX68000である」を読んでください。構造体や共用体などSmall-Cで省略された部分以外は，たいてい当てはまるはずです。また，ライブラリやSmall-C特有の機能などについては，参考文献2やSmall-C Ver. 2.7に付属していたドキュメントファイルを読んでいただくといいでしょう。

では，簡単なサンプルプログラムを作ってみましょう。いろいろ考えた末，Human上でいうところのDUMPコマンドを作ってみます(リスト1)。中身は簡単なのですが，ファイルの操作など，必要な情報は含んでいます。これからSmall-Cでプログラムを作ろうと考えている人は，参考にしてください。また，マシン語で同様のプログラムを作る場合を考えて，いかにC言語が便利であるか比べてみるのもいいでしょう。

### ・dump.Cの使い方

　# dump: ［ファイルネーム］

機能：［ファイルネーム］で指定されたファイルの内容を16進数で表示し，さらにASCIIダンプを行う。

### ・プログラムの説明

**1〜3行**　注釈です。

**5行**　スタンダードI/OヘッダファイルのSTDIO.Hを読み込んでいます。

**7行**　関数mainとコマンド定義をする。

**9〜12行**　使用する各変数を宣言する。その中身は 表1に示すとおりです。

**14行**　変数adrsに0を代入する。

**16行**　コマンドラインの第1パラメータ(argv［1］にその文字列が入っています)を入力ファイル名と見なして，ファイルを読み出し専用ファイルとして("r")オープンする。そして，このファイルの認識番号(ファイルディスクリプタ)をfd1に代入します。ただし，このときファイルのオープンに失敗していれば(ディスケットがさっていない，指定されたファイルが存在しないなど)，この値は0になります。通常，ファイルディスクリプタというのは，構造体(たいていFILEという名前になっている)へのポインタなのですがSmall-Cの場合，ファイルディスクリプタはBASICのように整数値になります。

**18〜21行**　もし，ファイルのオープンに失敗していたら画面に，

　file can't open

と表示して，プログラムの実行を中止する。

**23行**　ファイルの未読み取り部分があるかぎり，43行までの間をループをする。ファイルの内容をすべて読み終えたら，44行へ制御を移し，ファイルをクローズしたあとプログラムの実行を終了します。feof(fd)とは，fdで示されるファイルをすべて読み終えるとnullでない値を返す関数です。

**24行**　ここからバイナリダンプを行います，という注釈。

**25行**　変数adrsの内容を16進4桁で(%04x)表示する。

**26行**　0にセットされた変数iが(i=0)，1ずつ値を増加させて(i++)，iの値が16未満の間は31行までのループを繰り返します。つまり，31行までのループを16回繰り返していく，ということです(ただし，読み込み用のデータがある限りです)。

**27行**　ファイルから1バイト読み込みます。

**28行**　ところが，その値が1バイトの範囲でなく，EOF(End Of File，値としては−1)だと，何かよくないことが起こっているので，急いでループを抜け出す(break文)。

**29行**　あとでASCIIダンプをするときのために，i番目のデータを配列dataのi番目の要素として保存しておく。

**30行**　変数cの値を16進2桁で(%02x)表示を行います。

**32行**　画面にスペースを2つ表示する。普通，(16−i)＊3は0ですが28行でループを抜け出したときには，本来数字が表示されているはずのスペースを確保します(バイナリダンプとASCIIダンプとの境界用)。

**34行**　ここから，ASCIIダンプを行います，という注釈。

**35行**　26行と同様に40行までのループをi回繰り返します。つまり，バイナリダンプした分だけ，ASCIIダンプを行うのです。以下，基本的にはバイナリダンプのときと同じですが，データはファイルから持ってくるのではなく，配列のdataから持ってきます。また，キャラクタコード$20_H$未満のコードは，コントロールコードなので，画面に表示できません。代わりに"."を表示するようにしています(36〜39行)。

**41行**　変数adrsの値を16増やします。

**42行**　改行。

**43行**　ここまでの処理が終わるとまた23行まで処理が戻ります。もしも，ここでファイルがすべて読み終わっているなら，ループせずにファイルをクローズしてプログラムの実行を終了します。

**45行**　メインルーチンの終わりです。

**47行**　関数space の定義(この関数は引数として，整数値を1個取ります)。この関数の内容は，「引数cnt+3個のスペースを画面に表示する」です。

**48行**　41行と同様に，変数cntの値を3増やすというC言語独特の演算子です。

**49〜50行**　cntの値をループを1回まわるたびに1ずつ減らしていき(cnt−−)，0になるまで画面にスペースを出力し続ける(putchar(' '))。

### ・コンパイル方法と実行方法

以前から，ずっと同じようなことを書いている気がしますが，念のために復習しておきましょう。基本的には，未拡張のSWORDシステムを対象にしますので，拡張している方は各自アレンジしてください。Cコンパイラを，

　#L CC

でロードし，

　#J3000 dump −M −A −P −O −I

で，ソースファイルdump.Cをコンパイルします。コンパイルが終了したら，ディレクトリを取ってアセンブラファイルdump.ASMが，作成されているかを確認してください。次にそのファイルをWZDを用いてアセンブルします。

　#L WZD

**表1　サンプルプログラム1の変数表**

| | |
|---|---|
| fd： | ファイルディスクリプタ |
| i： | ループカウンタ用変数 |
| c： | データの一時待避用変数 |
| data： | データの一時待避用変数 |
| adrs： | アドレスを記憶する変数 |

#J3000 =dump

最後にライブラリファイルclib.LIBとリンクして，実行形式のファイルを作成します（各スイッチの意味は1991年8月号を参照してください）。

#L WLK

#J3000 dump,clib/s,dump/n

これで，dump.OBJというファイルが作成されていますので，試しに実行してみましょう。

#L dump.OBJ

#J31E5 dump.C

31E5は実行開始アドレスで，このアドレスはSmall-Cのバージョン，入手経路によって多少異なるかと思います。どうですか？ 先ほど打ち込んだdump.Cの内容が右端に表示されているでしょう。中央部に表示されている16進の数字が，dump.Cの16進ダンプです。

## サンプルプログラム2

今度は，Human上でいうところのCOPYコマンドを作ってみます。HumanのCOPYコマンドに比べて，ワイルドカードが使えない，ASC属性のファイルのみしか扱えない，と若干の制約があります。

中身はいたって簡単です。ファイルを読み込み用，書き出し用と2つオープンして，読み込み用ファイルを片っ端から読み込んでいき，データが続くかぎり書き込み用ファイルに書き出していくだけです（リスト2）。

**・copy.Cの使い方**

# dump: ［ファイルネーム1］［ファイルネーム2］

機能：［ファイルネーム1］で指定されたファイルの内容を［ファイルネーム2］で指定されたファイルにコピーする。

**・プログラムの解説**

**1～3行** 注釈です。

**表2 サンプルプログラム2の変数表**

| | |
|---|---|
| fd1： | 読み込みファイル用<br>ファイルディスクリプタ |
| fd2： | 書き込みファイル用<br>ファイルディスクリプタ |
| c ： | データの一時待避用変数 |

**5行** スタンダードI/OヘッダファイルのSTDIO.Hを読み込みます。

**7行** 関数mainとコマンドの定義をします。

**9～10行** 使用する各変数を宣言しています。その中身は表2に示すとおりになっています。

**12行** コマンドラインの第1パラメータ(argv［1］にその文字列が入っています)を送り元のファイル名と見なして，ファイルを読み出し専用ファイルとして("r")オープンします。

**13行** コマンドラインの第2パラメータ(argv［2］にその文字列が入っています)を送り先のファイル名と見なして，ファイルを書き込み専用ファイルとして("w")オープンします。

**15行** もしも，どちらかのファイルがオープンに失敗していたらエラー処理ルーチンへいきます（関数をコールしていますが，関数error( )からは，戻ってこないので，goto文と同じような結果になります)。C言語では，どちらかの条件が成り立っていたらというときには論理演算子“||”を使います。また，ともに成り立っていたら，というときは“&&”を使います。

**17行** 読み込み用ファイルの未読み取り部分があるかぎり，22行までの間をループします。ファイルの内容をすべて読み終えたら23行へ制御を移してファイルをクローズしてからプログラムの実行を終了します。

**18行** ファイルから1バイト読み込んで，その値を変数cに代入する。

**19行** 変数cの値が1バイトの範囲でなく，EOFだと，何かよくないことが起こっているので，急いでループを抜け出すのは，サンプルプログラム1と同じ。

**20行** 変数cの値を書き込み用ファイルに出力する。このとき，関数putcは書き込みに成功すると，戻り値としてcの値をそのまま返しますが，なんらかの理由で書き込みに失敗すると，戻り値としてEOFを返します。もしも，戻り値としてEOFが返されたときは，ループを抜け出します（21行)。

**22行** ここまでの処理が終わると17行まで処理が戻ります。ここでファイルがすべて読み終わっているなら，ループせずにファイルをクローズして(23～24行)，プログラムの実行を終了します。

**25行** メインルーチンの終わりです。

**27行** 関数errorの定義。メインループ中にファイル関係のエラーが発生した場合にここへ飛んできます。

**28行** 画面に，

file can't open

と出力します。

**29行** プログラムの実行を中止し，SWORDのコマンドモードに戻ります。

**・実行方法**

サンプルプログラム1と同様にして，コンパイル/リンクを行います。無事，コンパイルが成功したら，さっそく実行してみましょう。カレントドライブに先ほどの，copy.Cがあることを確認してください。そうしたら，

#L copy.OBJ

#J3AA5 copy.C test.C

と実行すると，あら不思議，test.Cというファイルが作成されていますね。試しに，エディタでtest.Cの内容をのぞいてみて，copy.Cと同じ内容であることを確かめておきましょう。

## さて，来月号は？

サンプルプログラム1，2ともに，ASCファイルはオープンできますが，BINファイルはオープンすることができません。これは，rdrtl.ASMをちょこちょこっと改造すればできるようになるので，機会があればなんとかしたいですね。

ということろで，今回はC言語の基本的な説明をさせてもらいました。まったくC言語を知らない人を対象に書いてきたので，すでに知っている人には，ものたりなかったかもしれませんね。そんな人は，サンプルプログラムをいじりまわしてS-OSとSmall-Cの関係を確かめるなり，もっと実用になるような改造をしてみるといいでしょう。

そして，来月号ではちゃんとSmall-Cの特徴を紹介しながら一般的なC言語との違い，その違いを吸収するための具体的な手法を見ていきます。冒頭でも述べているように，省略された機能が多少あります。フルセットのC言語に慣れている人は，いままで使えたものが使えないことで戸惑うことでしょう。そこでは，僕なりの解決方法

を示していこうと思っています。

開発用言語としてのC言語を有効に活用してもらうために，読者の皆さんが疑問に思ったこと，よくわからないことなどがあったら編集室までお便りください。リアルタイム，とはいきませんができるかぎりサポートしていきます。
質問のほかにもSmall-Cを活用するためのアイデア，プログラムを期待しています。

S-OSの世界でもC言語が使えるようになったのですから，やはりほかとは違ったS-OSらしいC言語にしていきたいもの。

機能拡張するにもいままでのS-OSの資産を使えば，それれに対応するライブラリの制作だけですみます。また，新しく拡張パッケージを作るのもいいでしょう。Small-C用のライブラリつきでね。

そして，ライブラリの追加によって拡張していくC言語の特徴をフルに生かして，何か面白いプログラムを作ってみようではありませんか。

では，来月号の応用編をお楽しみに。


**参考文献**

1) DDJ ツールブック DDJ編集部編 阿部尚子訳 工学社

2) プログラミング言語C B.W.Kernighan, D.M. Ritchie 石田晴久訳 共立出版社


## 今月のバグ出し

結構，慎重に移植したつもりだったのですが，まだまだ，バグがあったようです。というわけで，埼玉県八潮市の小原貴志さんからの指摘です。

・#include分において，ファイル名を<，>，"，で囲むとError になる。

せっかくオリジナルのバグを見つけて取り除いたつもりだったのに，かえってエンバグしてしまいました。

・DS命令の出力がおかしい。

WZDでは，オペランドは行の1文字目から始めてはいけないというルールがありました（1文字目から書くのを許されるのはコメントとラベルだけです。自分で決めておいて忘れるとはナサケナイ）。

・ファイル名に拡張子をつけたときの動作がおかしい。

たとえば，

CC: test.C −m −a −p −o −i

とするとエラーメッセージを出すわけでもなく，変なファイルを作成してしまいました。今回のデバッグでは，このような指定をすると出力は標準出力にするようにしますので，リダイレクトでアセンブラファイルを指定してください。普通は，−oオプションなどで出力ファイルの指定ができるのですが，結局同じことなのでやめておきました。

・エラーを示す記号がおかしい。

たとえば，

a = 1 ＊＊ 2;

をコンパイルしようとすると，

/¥　　Error

のような記号でエラーの箇所が指摘されてしまいます。これは海の向こうのアメリカで，"¥"という記号の代わりに"\"という記号を持っていることによります。

アメリカやヨーロッパのコンピュータでは"/\"のように矢印になります。私は，この記号を見るたびに，「あぁ，外国からはるばる海を渡ってきたプログラムを使っているんだなぁ」と，感慨に耽っていたものですから，直さなきゃいけないことすら忘れていました。対処法は簡単です。"¥"の代わりに別のそれらしい記号"1"かなにかにすればいいのです。そうすれば，ちゃんと，矢印のようになります。気分的な問題で実害はないのですが訂正しておいてください。

あと，一部の機種で関数pollにおいて，ctrl＋cキーが効かないという症状があるようです。これはSWORD内でキャラクタコード$03_H$がマスクされているため，このような事態が起こってしまいます。マスクを外すか，6月号のリスト8，poll.macの2行目を

CTRL_C　EQU　　1BH

に変更してください。この変更によってSHIFT＋BREAKキーがCTRL＋Cキーの代わりになります。

デバッグ方法は，ソースをお持ちの方であればリスト3に示す関数の訂正を行ってください。ところが，6月号で配布したディスクには，私の手違いでコンパイラ本体のソースが入っていませんでした。

今回のデバッグをしてくださった小原さんはなんと，コンパイラを逆アセンブルしてデバッグしてくださったようです。重ね重ね申し訳ありません。また小原さんはMAGIC，TURTLE，SOROBANのライブラリも作成しようとしているようです。期待していますからがんばってくださいね。

で，ソースをお持ちでない方は，リスト4，5を入力してそれぞれ，SC_DEBUG DATA.OBJ，SC_DEBUG.OBJのファイル名でセーブを行ったあと，以下の手順でパッチを当ててください。

まず，SC.COM，SC_DEBUG DATA.OBJ，SC_DEBUG.OBJをロードしてコマンドラインから，

#JB000

と打ち込んで，パッチ当てプログラムを実行します。実行が終わったら，

#S SC1.COM:3000：A735：3000

で，セーブをしてください。これで訂正は終わります。

### リスト1

```
==================  dump.C  ==================
     1: /* Dump Program
     2: ** By T.Ishigami '91/08/10
     3: */
     4:
     5: #include <stdio.h>
     6:
     7: main(argc, argv) int argc; char **argv; {
     8:
     9:   int fd;
    10:   int i, j, c;
    11:   char data[16];
    12:   int  adrs;
    13:
    14:   adrs = 0;
    15:
    16:   fd = fopen(argv[1],"r");
    17:
    18:   if(fd == 0) {
    19:     puts("file can't open¥n");
    20:     exit();
    21:   }
    22:
    23:   while(feof(fd) == NULL) {
    24: /* Binary Dump */
    25:     printf("%04x: ",adrs);
    26:     for(i = 0; i < 16; i++) {
    27:       c = getc(fd);
    28:       if(c == EOF) break;
    29:       data[i] = c;
    30:       printf("%02x ",c);
    31:     }
    32:     space((16-i)*3);
    33:
    34: /* Ascii dump */
    35:     for(j = 0; j < i; j++) {
    36:       if(data[j] < 0x20)
    37:         putchar('.');
    38:       else
    39:         putchar(data[j]);
    40:     }
    41:     adrs += 16;
    42:     putchar('¥n');
    43:   }
    44:   fclose(fd);
    45: }
    46:
    47: space(cnt) int cnt; {
    48:   cnt += 3;
    49:   while(cnt--)
    50:     putchar(' ');
    51: }
```

## リスト2

```
==================  copy.C  ==================
 1: /* Copy Program
 2: ** By T.Ishigami '91/08/11
 3: */
 4:
 5: #include <stdio.h>
 6:
 7: main(argc, argv) int argc; char **argv; {
 8:
 9:     int fd1,fd2;
10:     int c;
11:
12:     fd1 = fopen(argv[1],"r");
13:     fd2 = fopen(argv[2],"w");
14:
15:     if(fd1 == 0 || fd2 ==0) error();
16:
17:     while(feof(fd1) == NULL) {
18:       c = getc(fd1);
19:       if(c == EOF) break;
20:       c = putc(c,fd2);
21:       if(c == EOF) break;
22:     }
23:     fclose(fd1);
24:     fclose(fd2);
25: }
26:
27: error() {
28:     puts("file can't open¥n");
29:     exit();
30: }
```

## リスト3

```
==================  dumpzero.c  ==================
 1: /*
 2: **        ( cc11.c )
 3: /*
 4: ** dump zeroes for default initial values
 5: */
 6: dumpzero(size, count) int size, count; {
 7:   int j;
 8:
 9:   if(!iflag){          /* fas 2.2 */
10:     if(count > 0){
11:     ot(" DS ");          /* Debugged '91 Jul.11th */
12:     outdec(size * count);
13:     nl();
14:     }
15:   }else{
16:     while (count > 0) {
17:       poll(1); /* allow program interruption */
18:       defstorage(size);
19:       j=30;
20:       while(j--) {
21:         outdec(0);
22:         if ((--count <= 0)|(j==0)) {
23:           nl();
24:           break;
25:           }
26:         outbyte(',');
27:         }
28:       }
29:     }
30:   }
31:
==================  openfile.c  ==================
 1: /*
 2: **          (cc11.c)
 3: */
 4:
 5: /*
 6: ** input and output file opens (変更あり)
 7: */
 8: openfile() {    /* entire function revised *//*39*/
 9:   char fn[20];  /* file name buffer 2 + 13 + 1 + 3 + 1*/
10:   char *cmdptr; /* command line pointor */
11:   int i, ext;
12:   input=EOF;
13:   line = pline;
14:
15:   while(++filearg < argcs) {
16:         cmdptr = argvs[filearg];
17:         if(cmdptr[0]=='-') continue;
18:
19:         ext = NO;
20:         i = 0;
21:
22:         while(cmdptr[i] && i < 15) {
23:                 if(cmdptr[i] == '.') {
24:                         ext = YES;
25: /*                      break;  Debugged '91 Jul.11th */
26:                 }
27:                 fn[i] = cmdptr[i++];
28:         }
29:         fn[i] = 0;          /* Debugged '91 Jul.11th */
30:         if(!ext) strcpy(fn + i, ".C");
31:
32:         input = mustopen(fn, "r");
33:
34:         if(!files && isatty(stdout)) {
35:                 strcpy(fn + i, ".ASM");
36:                 output = mustopen(fn, "w");
37:         }
38:         files=YES;
39:         kill();
40:         return;
41:   }
42:   if(files++) eof=YES;
43:   else input=stdin;
44:   kill();
45:   }
==================  doinclude.c  ==================
 1: /*
 2: **        ( cc12.c )
 3: */
 4: /*
 5: ** open an include file
 6: ** (変更あり)
 7: */
 8: doinclude()  {
 9:   char c, *fname, buff[17];
10:   int i;
11:
12:   blanks();          /* skip over to name */
13:   /*
14:    * added code to handle include
15:    * filename in quotes or brackets
16:    * 4/5/83 br
17:    */
18:   if ((*lptr == '"') | (*lptr == '<')) {
19:     lptr++;                 /* Debugged '91 Sep 20th */
20:     fname = buff;
21:     for(i = 0; i < 16; i++) {
22:       c = *lptr++;
23:       if ((c == '"') | (c == '>')) break;
24:       *fname++ = c;
25:     }
26:     *fname = '¥0';
27:     fname = buff;
28:   }
29:   else
30:   fname = lptr;  /* no'"'or'<' (original convention)*/
31:
32:                                        /*fas 2.7*/
33:    if(inclevel <= 5){
34:      if((input2[++inclevel]=fopen(fname,"r"))==NULL) {
35:       input2[inclevel--]= EOF;
36:       error("open failure on include file");
37:      }
38:    }else{
39:      error("maximum include nesting reached");
40:    }
41:
42:   kill();          /* clear rest of line */
43:       /* so next read will come from */
44:       /* new file (if open) */
45:   }
46:
==================  errout.c  ==================
 1: /*
 2: **        ( cc22.c )
 3: */
 4: errout(msg, fp) char msg[]; int fp; {
 5:   int k; k=line+2;
 6:   while(k++ <= lptr) cout(' ', fp);
 7:   lout("/1", fp);
 8:   sout("**** ", fp); lout(msg, fp);
 9:   }
```

## リスト4

```
3F4C 21 00 00 39 EB 21 00 00 : 66
3F54 CD 9E A6 21 02 00 39 EB : 58
3F5C 21 00 00 CD 9E A6 21 04 : 57
3F64 00 CD 5B A6 EB C1 E1 E5 : 40
3F6C C5 CD 4A A6 EB 21 2E 00 : BC
3F74 CD B9 A6 7C B5 CA 87 3F : ED
3F7C 21 00 00 39 EB 21 01 00 : 67
3F84 CD 9E A6 21 06 00 39 EB : 5C
3F8C C1 E1 E5 C5 19 E5 21 06 : 71
3F94 00 CD 5B A6 EB 21 04 00 : DE
3F9C CD 5B A6 CD 4A A6 D1 7D : D9
3FA4 12 21 04 00 CD 5B A6 E5 : EA
3FAC 21 04 00 CD 8E A6 2B D1 : 22
```

```
3FB4 CD 4A A6 7C B5 CA DC 3F : D3
3FBC C1 D1 D5 C5 21 0F 00 CD : 29
3FC4 D9 A6 7C B5 CA DC 3F C3 : 58
---------------------------------
SUM: B7 7E 78 44 50 F6 0C 06 4221

3FCC 62 3F 20 44 53 20 00 00 : 78
3FD4 00 00 00 00 00 00 00 00 : 00
3FDC E1 E5 CD.28 A7 7C B5 CA : 5D
3FE4 FB 3F 21 06 00 39 EB C1 : 46
3FEC E1 E5 C5 19 2B          : CF
---------------------------------
SUM: 1F 48 D3 8B 25 D5 A0 8B 8AB9
```

## リスト5

```
B000 AF 32 F3 42 21 CE 3F 22 : 66
B008 B6 3C 21 39 70 36 2F 23 : 44
B010 36 6C C9                : 6B
---------------------------------
SUM: 9B DA DD 7B 91 04 6E 45 31D3
```

## 全機種共通システムインデックス

■85年6月号
序論　共通化の試み
第1部　S-OS"MACE"
第2部　Lisp-85インタプリタ
第3部　チェックサムプログラム
■85年7月号
第4部　マシン語プログラム開発入門
第5部　エディタアセンブラZEDA
第6部　デバッグツールZAID
■85年8月号
第7部　ゲーム開発パッケージBEMS
第8部　ソースジェネレータZING
■85年9月号
インタラプト　S-OS番外地
第9部　マシン語入力ツールMACINTO-S
第10部　Lisp-85入門(1)
■85年10月号
第11部　仮想マシンCAP-X85
連載　Lisp-85入門(2)
■85年11月号
連載　Lisp-85入門(3)
■85年12月号
第12部　Prolog-85発表
■86年1月号
第13部　リロケータブルのお話
第14部　FM音源サウンドエディタ
■86年2月号
第15部　S-OS"SWORD"
第16部　Prolog-85入門(1)
■86年3月号
第17部　magiFORTH発表
連載　Prolog-85入門(2)
■86年4月号
第18部　思考ゲームJEWEL
第19部　LIFE GAME
連載　基礎からのmagiFORTH
連載　Prolog-85入門(3)
■86年5月号
第20部　スクリーンエディタE-MATE
連載　実戦演習magiFORTH
■86年6月号
第21部　Z80TRACER
第22部　magiFORTH TRACER
第23部　ディスクダンプ&エディタ
第24部　"SWORD" 2000 QD
連載　対話で学ぶ magiFORTH
特別付録　PC-8801版S-OS"SWORD"
■86年7月号
第25部　FM音源ミュージックシステム
付録　FM音源ボードの製作
連載　計算力アップのmagiFORTH
特別付録　SMC-777版S-OS"SWORD"
■86年8月号
第26部　対局五目並べ
第27部　MZ-2500版S-OS"SWORD"
■86年9月号
第28部　FuzzyBASIC発表
連載　明日に向かって magiFORTH
■86年10月号
第29部　ちょっと便利な拡張プログラム
第30部　ディスクモニタ DREAM
第31部　FuzzyBASIC 料理法<1>
■86年11月号
第32部　パズルゲーム HOTTAN
第33部　MAZE in MAZE
連載　FuzzyBASIC 料理法<2>
■86年12月号
第34部　CASL & COMET
連載　FuzzyBASIC 料理法<3>
■87年1月号
第35部　マシン語入力ツールMACINTO-C
連載　FuzzyBASIC 料理法<4>
■87年2月号
第36部　アドベンチャーゲーム MARMALADE
第37部　テキアペ作成ツール CONTEX
■87年3月号
第38部　魔法使いはアニメがお好き
第39部　アニメーションツール MAGE
付録　"SWORD" 再掲載と MAGIC の標準化
■87年4月号
第40部　INVADER GAME
第41部　TANGERINE
■87年5月号
第42部　S-OS"SWORD" 変身セット
第43部　MZ-700用 "SWORD" を QD 対応に
■87年6月号
インタラプト　コンパイラ物語
第44部　FuzzyBASIC コンパイラ
第45部　エディタアセンブラ ZEDA-3
■87年7月号
第46部　STORY MASTER
■87年8月号
第47部　パズルゲーム碁石拾い
第48部　漢字出力パッケージ JACKWRITE
特別付録　FM-7/77版 S-OS"SWORD"
■87年9月号
第49部　リロケータブル逆アセンブラ Inside-R
特別付録　PC-8001/8801 版 S-OS"SWORD"
■87年10月号
第50部　tiny CORE WARS
第51部　FuzzyBASIC コンパイラの拡張
第52部　X1turbo 版 S-OS"SWORD"
■87年11月号
序論　神話のなかのマイクロコンピュータ
付録　S-OS の仲間たち
第53部　もうひとつの FuzzyBASIC 入門
第54部　ファイルアロケータ&ローダ
インタラプト　S-OS こちら集中治療室
第55部　BACK GAMMON
■87年12月号
第56部　タートルグラフィックパッケージTURTLE
第57部　X1turbo 版 "SWORD" アフターケア ラインプリントルーチン
特別付録　PASOPIA7 版 S-OS"SWORD"
■88年1月号
第58部　FuzzyBASIC コンパイラ・奥村版
付録　石上版コンパイラ拡張部の修正
■88年2月号
第59部　シューティングゲーム ELFES
■88年3月号
第60部　構造型コンパイラ言語 SLANG
■88年4月号
第61部　デバッギングツール TRADE
第62部　シミュレーションウォーゲーム WALRUS
■88年5月号
第63部　シューティングゲーム ELFES II
第64部　地底最大の作戦
■88年6月号
第65部　構造化言語 SLANG 入門(1)
第66部　Lisp-85 用 NAMPA シミュレーション
■88年7月号
第67部　マルチウィンドウドライバ MW-1
連載　構造化言語 SLANG 入門(2)
■88年8月号
第68部　マルチウィンドウエディタ WINER
■88年9月号
第69部　超小型エディタ TED-750
第70部　アフターケア WINER の拡張
■88年10月号
第71部　SLANG 用ファイル入出力ライブラリ
第72部　シューティングゲーム MANKAI
■88年11月号
第73部　シューティングゲーム ELFES Ⅳ
■88年12月号
第74部　ソースジェネレータ SOURCERY
■89年1月号
第75部　パズルゲーム LAST ONE
第76部　ブロックゲーム FLICK
■89年2月号
第77部　高速エディタアセンブラ REDA
特別付録　X1版 S-OS"SWORD"<再掲載>
■89年3月号
第78部　Z80用浮動小数点演算パッケージSOROBAN
■89年4月号
第79部　SLANG 用実数演算ライブラリ
■89年5月号
第80部　ソースジェネレータ RING
■89年6月号
第81部　超小型コンパイラTTC
■89年7月号
第82部　TTC用パズルゲーム　TICBAN
■89年8月号
第83部　CP/M用ファイルコンバータ
■89年9月号
第84部　生物進化シミュレーションBUGS
■89年10月号
第85部　小型インタプリタ言語TTI
■89年11月号
第86部　TTI用パズルゲーム　PUSH BON!
■89年12月号
第87部　SLANG用リダイレクションライブラリ DIO. LIB
■90年1月号
第88部　SLANG用ゲームWORM KUN
特別付録　再掲載SLANGコンパイラ
■90年2月号
第89部　超小型コンパイラTTC++
■90年3月号
第90部　超多機能アセンブラOHM-Z80
■90年4月号
第91部　ファジィコンピュータシミュレーションI-MY
■90年5月号
第92部　インタプリタ言語STACK
■90年6月号
第93部　リロケータブルフォーマットの取り決め
第94部　STACK用ゲーム SQUASH！
第95部　X68000対応S-OS "SWORD"
特別付録　PC-286対応S-OS "SWORD"
■90年7月号
第96部　リロケータブルアセンブラWZD
■90年8月号
第97部　リンカWLK
■90年9月号
第98部　BILLIARDS
■90年10月号
第99部　ライブラリアンWLB
■90年11月号
第100部　タブコード対応エディタEDC-T
■90年12月号
第101部　STACKコンパイラ
■91年1月号
第102部　ブロックアクションゲーム COLUMNS
■91年2月号
第103部　ダイスゲームKISMET
■91年3月号
第104部　アクションゲームMUD BALLIN'
■91年4月号
第105部　SLANG用カードゲームDOBON
■91年5月号
第106部　実数型コンパイラ言語REAL
■91年6月号
第107部　Small-C処理系の移植
■91年7月号
第108部　REALソースリスト編
■91年8月号
第109部　Small-Cライブラリの移植
■91年9月号
第110部　SLANG用NEWファイル出力ライブラリ

＊以上のアプリケーションは，基本システムであるS-OS "MACE" または S-OS "SWORD" がないと動作しませんのでご注意ください。

INDEX

特別レポート・ソフトウェア市場を分析する

# X68000ゲームソフトのゆくえ

編集部

**X68000のゲームソフト市場に不穏な動きが起こっている。ソフトハウスがユーザーを糾弾。違法コピーの横行とそれによる市場低迷を訴える記事が続発。いったいどういうことなのだろう。市場データをもとに分析してみよう。**

X68000が発売されて4年半がたつ。出荷台数も13万台を超えている。正確なユーザー数はわからないが，10万人前後のユーザーが実働していると推定される。パソコンの世界では一応10万台を超えれば市場的にも認知されたことになるそうだが，その意味ではX68000も市販ソフトが安定して売れる土台が出来てきたといってよい。

さて，X68000といえば当初から卓越したグラフィック機能，サウンド機能でアーケードゲームの移植レベルも高く，熱心なユーザーの期待を集める希有なマシンであった。また，自分たちが育てるマシンという意識を持ったユーザーが多く，ソフトの売れ行きもハードの台数からは考えられないほど好調で「ユーザーがソフトを買い支えている」といった話もある程度の真実味をもっていた。ユーザー自身もそのことを誇りに感じていたと思う。

ところがここにきて，X68000のソフトがピンチという騒ぎが起きている。それもユーザーに対する警告といったかたちで現れてきている。

## ソフトハウスがX68000離れ？

内容は，ユーザー間での違法コピーのために，ソフトの販売数が減少し，ソフトハウスがX68000から撤退するかもしれないというものだ。これらの話はソフトハウスの広告や，ソフトのデモ画面，そしてソフトハウスの声を代弁する事情通の人間をとおしてユーザーに向けられている。

たとえば，X68000のゲームメーカーとして注目度ナンバーワンのズーム。その「ありがとう広告・1991・」(ログイン No.14に掲載)では，お茶目なメッセージのなかに違法コピーの件を取り上げ「実際その辺のせいで今後事としだいによっては他機種に移らないと……」といったことが書かれており，大きな反響を巻き起こしている(良きにつけ悪しきにつけズームの影響力は大きいのだ)。

また，このズームの広告に呼応してＳＰＳも，マイコンBASIC Magazine 9月号のＳＰＳスピリッツのコーナーで「我が社も同じ考えです。(中略)わかっていただけましたよね？」とユーザーに釘をさすという始末だ。

ソフトハウスからこうした声が出てくるのは，ユーザー数が増えているのにソフトの売り上げが伸びず，むしろ減少しているという状況が根本にあるようだ。それがどうしたわけか違法コピーの問題にすりかわってしまった。

そんな馬鹿な，と疑問に感じる人も多いだろう。そんなに違法コピーは多いのか。それによって，ソフトが売れていないのか。そもそもソフトが売れていないというのは本当なのか。

## 血迷った事情通

迷惑な話だが，ズームの広告が載った翌15号ではX68000とはなんの関わりもない双葉社の広告に「X68000ソフトの1タイトル当たりの実売本数が激減している(全盛時の7割減??)」という寄稿があり，X68000のソフトがなくなる！ と騒ぎを助長させている。

もっともっとひどい話もある。X68000アイドル(？)の山下章氏がマイコンBASIC Magazine 9月号で同様のことを語っているが，パロディウスだ！ がユーザー数の「4分の1の本数も売れていないらしい」といい，「半分以上のユーザーがソフトをもっているにちがいないのに」とまでいっている。さらに，3年前は4万本売れたのに，今は2万本しか売れない，と続け違法コピーのために泣きを見るのはソフトハウスだと説いている。

ここまで馬鹿な話になると，真に受ける人も少なくはなるだろうが，それにしてもひどい。

パロディウスだ！ に関しては，やはり多くのX68000ユーザーに買ってもらいたいソフトである。技術力のあるメーカーが本気で作ればこれだけのことができるというお手本のようなソフトだからだ。実は店頭デモの段階ではかなりスピード的に遅く処理の重さが目立っていた。完成版では見事に克服されていたのだが，店頭デモを見た人やゲーム雑誌で途中バージョンの情報を目にした人も多かったはずだ。Oh!Xでパロディウスだ！ を3カ月連続で紹介したのも，前評判でパロディウスだ！ のすごさを知らない人がいたら困ると思ったからだ。

もちろん，そうした事情がなくても2万本というのはかなり限界に近い数字だろう。この春のラインアップを見ても1本のソフトを4人に1人が買うというのは難しい。また，ユーザーの半分以上というのは，仮にパロディウスだ！ がPDSだったとしてもOh!Xの付録にでもしない限り不可能な数字だ。だいたい，3年前には4万本どころかユーザー数だって4万人もいなかったはずだ(シャープによると3年前の9月の時点で出荷台数は3万5千台だ)。

ところで，X68000ユーザーにはコピーユーザーが多いと主張する人の根拠は概ね次の理由だ。まず，X68000も発売から4年がたっているということ，年齢層が若くマニアックなユーザーが多いこと，そしてユーザー同士の情報交換が緊密になされているといったことだ。

実際問題としては，違法コピーの実態はわからないというのが正しい。

もちろん違法コピーがあるのは事実だろう。いや，ないわけはない。が，レンタルショップがなくなったいま，個人ベースで市場の変化を引き起こすほど増えているとは考えにくい。ネットワークを利用したコピー業者があるとしても，それはX68000に限ったことではない。

いずれにしても，データ的な裏付けがない以上，ソフトの不振をコピーユーザーのせいにするのはあまりにも問題だ。

実際，これらの広告や記事を読んでショックを受けたユーザーは多いだろう。真面目にソフトを買って遊んでいるユーザーがこんなことをいわれて傷つかないはずがない。そしてもっと怖いのは，違法コピーの件が真実だろうがデタラメだろうが，とにかくもっと買ってくれなければもうソフトを作らないということだからだ。目的がコピー防止運動でもユーザーにとっては脅迫になる。

## 市場の規模を確認しよう

ともかく，実際にソフトの売れ行きに陰りがあるとしたらそれは違法コピーのせいでなくとも問題がある。ソフトが売れないマシンはかつて消えていった多くのマシンと同じ運命を辿ることは間違いない。

そこで，ソフト市場についての実態を流通に詳しい関係者に聞いてみたところ「確かに，昨年後半あたりから1タイトルあたりの数が減少しているようです。でも，タイトル数がかなり増えていますからね。全体としてはハードの流れに応じて伸びているといえるでしょう」という答えが返ってきた。7割減なんてことはありえない。前作が1万本売れたのに新作が3000しか売れなかったら，よほど前作が不評を買ったかあるいは営業的な失敗だろう。

まずグラフ1がX68000の出荷台数とゲームソフトのタイトル数の推移である。そして表1がX68000用ゲームソフトのタイトル数を年度ごとにまとめたもの。データはソフトバンクの流通を通して出荷されたものにシャープ，電波新聞社，コンパックなどの製品を加えたもので，発売時期を確認したものだけでも240種（7月18日現在）はある。実際には，ブラザー工業のTAKERUで販売されているものやマイナーな美少女ソフトなどを加えるとこれよりもかなり多い。現時点で市場に出回っているゲームソフトは300種以上はあると思われる。

この表1をご覧になればわかるとおり，初代X68000が発売された1987年にはゲームソフトは僅か9作品しか発売されなかった。1988年が24種類，1989年が60種類で，ユーザー数の増加と共に順調に伸びている。ところが問題は昨年だ。なんと103種類ものゲームが発売されている（実際は140種近くあったのでは？）。しかもその7割方が昨年後半から年末にかけて集中しているのだ。

一方，グラフ1を見ると，1989年には4万台も増えているのに，1990年には2万5千台しか増えていない。これについてはある程度理由がはっきりしている。昨年の春に発売されたEXPERT II/PRO IIはSX-WINDOWがあったとはいえ，ハード的にはX68000のロゴが金バッジになっただけ。目玉のSUPER-HDは発売が遅れた。他のメーカーが高速化，小型化，低価格化を進めているなかにあって昨年のラインアップは商品的には弱かったといえるだろう。

こうしてソフトの購買意欲が高い新規ユーザーが少ないところに新作ソフトの急激な増加。これではいくらユーザーがソフトを買っても追いつかない。しかも新しくX68000を買ったユーザーが必ずしも新作ソフトしか買わないわけではなく，ソフトを選ぶ際には人気の高かった過去の作品だって選択の対象に入るわけだ。

## 価格の上昇も見逃せない

市場規模の大まかな推移を見てきたわけだが，ハードの出荷台数とソフトの種類以外にもうひとつ重要な要素がある。それはソフトの価格だ。これもデータを見てみよう。4年前の1987年に発売された9作品の平均価格は7,144円。今年発売された44作品の平均価格は8,482円。4年間で18.7%の上昇だ。前年比4.4%の上昇率ならそれほど上がっているとはいえないだろう。

が，ここにはちょっとした落とし穴がある。今年の作品44作品の内訳を見ると美少女ソフトが17作品，マップエディタが1作品あり，これらは比較的安価である。美少女ソフト17作品の平均価格は7,271円。これに対し一般のゲーム26作品の平均価格は9,415円もする。つまり，一般のゲームの場合は1本当たりの価格がこの4年間になんと31.8%も上昇しているわけだ。こうなると多少の影響はあるのではないだろうか。かつてはマイナーな感の強かったアダルトソフトも美少女ソフトと呼ばれるようになり，グラフィックの強いX68000にとっては有利なジャンルとなっている。値段の高い一般ソフトからユーザーが流れても不思議はない。

## ユーザー側の状況は

こうした市場の動きのなかでユーザー側のソフト購入状況はどうだろう。

Oh!X 6月号のハガキで「この1年間で買ったソフトは何本でしょう」という質問を載せたところ，平均で5.2本という結果が出た。このなかにはゲームだけでなくSX-WINDOWやXCなども含まれていると思うが，それはこの際問題ではない。これを

グラフ1　X68000出荷台数とゲームソフトの推移

表1　年度別新作ゲームタイトル

| 年度 | タイトル数 | 主なタイトル |
|---|---|---|
| 1987年 | 9種 | ゼビウス／スペースハリアー／マンハッタン・レクイエム |
| 1988年 | 24種 | 源平討魔伝／ドラゴンスピリット／ツインビー／沙羅曼蛇／リターン・オブ・イシター／琥珀色の遺言／A列車で行こうII／サンダーフォースII／テトリス |
| 1989年 | 60種 | アフターバーナー／ボスコニアン／ファンタジーゾーン／パックマニア／ニュージーランドストーリー／今夜も朝までPOWERFULまあじゃん2／スタークルーザー／R-TYPE／ジェノサイド／A-JAX／フラッピー2／メタルサイト／ナイトアームズ／サバッシュ |
| 1990年 | 103種 | ダンジョン・マスター／ポピュラス／ワンダラーズ・フロム・イース／グラナダ／サーク／ルーンワース／ワールドコート／三国志II／ガンシップ／シムシティー／ラグーン／遊撃王IIエアー・コンバット／ナイアス／ソル・フィース／エメラルド・ドラゴン／イメージファイト／ワールドスタジアム／銀河英雄伝説II／シュヴァルツシルト／スーパーハングオン／サンダーブレード／サイバリオン／バブルボブル／ギャラガ'88 |
| 1991年 | 44種<br>(7/18現在) | マーブルマッドネス／メルヘンメイズ／ノスタルジア／遙かなるオーガスタ／パロディウスだ！／ファランクス／スコルピウス／サイレントメビウス／黄金の羅針盤／イース |

年齢で3つのグループに分けると表2のようになる。18歳以下で4.2本,19～22歳で4.6本,23歳以上だと7.4本にもなる。また,年齢層を問わず3本と答えた人が最も多かった。

これだけを見てもソフトの市場が歪んだものであることがわかるだろう。大方のゲームが中高校生をターゲットにした作りになっているのに対し,数の上では社会人のほうが多く購入しているのだ。

「だって,値段が高すぎますよ。高校生が9,800円もするソフトを何本も買えるわけないじゃないですか」と西川善司氏はいつものようにソフトの価格に怒りをぶつける。

確かにそうかもしれない。5月号でも発表したが,昨年8月号のハガキで質問した「1カ月のおこづかい」は表3のとおり,3,000～5,000円を中心に山がある。これだと,年3本が精一杯だろう。中～高～大学生の場合は経済的理由から限界があるということだ。

一方,20,000円以上で再び多くなり25.4%となっているが,これは社会人なら不思議はない額だろう。ただし,社会人の場合でも年3本の人が最も多いのは変わらないというのが気になるところだ。

社会人の場合はある程度経済力があるから買う人はたくさん買って平均値を上げているが,多くの人が欲しいと思って買っているのはやはり3本くらいしかないというのが実態なのではないだろうか。これについて中野修一氏は,

「値段は高くてもいいんですが,はっきりいって全体的にもっとクオリティを上げてほしいですね。それにX68000はゲームマシンだと思っているソフトハウスが多いようですが,ゲームしかやらないユーザーはそんなに多くはないんですよ」と語っている。

## 誤解されたX68000像

X68000のソフトが急速に増えた理由は,X68000の市場が大きくなってきたこともあるが,PC-8801やMSXがいきづまってきたことが大きい。PC-9801だとアクションゲームなどはハード的に難しい面があるし,スーパーファミコンは20種くらいのソフトがしのぎを削る非常に厳しい世界だ。そこでゲームが売れるというX68000に期待が集まってきたのだろう。

実際,X68000はゲームマシンと思われがちだ。それもシューティングゲームが強いというのが定説になっている感がある。だからといって,ゲーム機やMSXにあるようなタイプのゲームをいくらX68000用にパワーアップしてもユーザーのすべての層にうけるわけではない。年齢層が違うというのが理由のひとつ。グラフ2の年齢分布を見ればわかる。18歳以下のユーザーが37%程度であることを考えれば,デザインにしてもシナリオにしても幼稚なものが売れないのは当然だ。

そしてX68000ユーザーはゲームを楽しんでいるが,ゲームマシンのユーザーとは指向性が違うということも押さえておく必要がある。X68000でこれまで最もヒット作を出しているのは電波新聞社だが,スペースハリアーやアフターバーナーにしても,ゲームの面白さと移植の完成度だけで売れたのではない。X68000というマシンで実現できる世界の素晴らしさを見せたことが大ヒットの最大の要因だといってよい。それに,競合するソフトがあまりない時代の話だ。

昨年は,海外ソフトの移植作品が人気を独占したが,X68000用で最も数多く売れたゲームソフトベスト5は表4のようになる(参考としてOh!XのGAME OF THE YEAR(ゲーム大賞)での順位を記してあ

**表2 この1年間に買ったソフトの本数**

| 年齢層 (人数) | 平均 |
|---|---|
| 23歳以上 (133人) | 7.4本 |
| 19～22歳 (183人) | 4.6本 |
| 18歳以下 (184人) | 4.2本 |
| 全体 (300人) | 5.2本 |

1991年6月号

**表3 1カ月のおこづかいは**

| 金額 | 割合 |
|---|---|
| 20,000円以上 | 25.4% |
| 15,000～20,000円 | 7.3% |
| 10,000～15,000円 | 3.4% |
| 5,000～10,000円 | 14.3% |
| 3,000～ 5,000円 | 18.5% |
| 3,000円以下 | 9.5% |
| 不定 | 15.1% |
| なし | 6.5% |

1990年8月号

**表4 1990年X68000ゲーム販売数ベスト5**

| 順位 | 作品名 | GAME OF THE YEAR |
|---|---|---|
| 1位 | シムシティー | 2 |
| 2位 | ラグーン | 4 |
| 3位 | ダンジョン・マスター | 1 |
| 4位 | ポピュラス | 3 |
| 5位 | 三国志II | 8 |

**グラフ2 X68000ユーザーの年齢分布**

る)。シューティングに強いX68000でもシムシティーの強さに変わりはない。というか，アクションゲームはひとつもベスト5に入っていない。ナイアス，ソル・フィースといった話題作ではなく，三国志IIが5位に入っている。ラグーンがなければまるでPC-9801のランキングのようだ。

その意味でいってもX68000のオリジナルソフトとしてラグーンの健闘は際立つ。が，そのズームも今年のファランクスでは，パロディウスだ！ に多少水をあけられたようだ。

2年前，ジェノサイドをひっさげて登場したズームは本当にかっこよかった。X68000ユーザーの心を一気につかんだという感じであった。ズーム自身がそれほど気に入ってないという2作目のラグーンが大ヒットしたのも前作の評価が高かったからだろう。逆に3作目のファランクスはラグーンほどは売れないだろうというのが編集部内外の予想であった。

しかし，ファランクスはクオリティも高く，なにより「頑張っているのが伝わる」素晴らしい出来栄えであった。パロディウスだ！ のあとで多少の不利はあったものの，ラグーンの内容に首をかしげたファンもズームの力を再評価したはずだ。ハードディスクへのインストールが可能なことを評価する人も多い。しかし，「ズームが違法コピーの横行を疑ったのは，プロテクトを弱くしたことが裏目に出たと思ったのでしょう」と指摘する人もいる。そのためか例の広告でユーザーからいらぬ不信感を買ってしまったのは残念だが，なんといってもズームはX68000のゲームを引っ張っていってもらいたいソフトハウスである。

## X68000ユーザーの生きる道

このように見てくるとX68000のソフトが売れなくなってきたという説の真相は，マーケティング不足による不適切な商品投入であると考えられる。だが，今回はそれ以上に状況を把握していない人が騒ぎすぎた。しかも，違法コピーという刺激的なネタだけに，情報が歪んだかたちで広まってしまった。

違法コピーの実態を正確につかんでいる人はいないだろうし，違法コピーの量とソフトの実売数の関係を正確に説明できる人もいない。なかには，「違法コピーがなくなれば，ソフトの価格は半分になる」と主張する人もいるが，そういった主張もこれまではあくまで「ひとつの主張」として扱われてきた。一方で，「コピーがなくなってもソフトの値段は下がらない」と主張する人もいるわけだ。

だが，今回の事件は，そういった判断とは無関係に違法コピー説が前提になってしまった。なぜなら，ソフトハウスにとっては違法コピーの実態よりもソフトが売れないことのほうが遙かに直接的な問題であるからだ。そして私たちユーザーにとってもそのことのほうが重要な問題といえるだろう。

コピーうんぬんよりも，ユーザー数が多い少ないよりも，ソフトの種類がどうのこうのよりも，ソフトのクオリティがどうしたこうしたよりも，実際問題として，思うようにソフト売れなくて困っているソフトハウスが多いとしたら，X68000ユーザーはどうしたらいいのだろう。理由はどうあれ利益が出なければ，次回作はない。それに，一度コピーユーザーが多いという不名誉な評判を立てられた以上，なんとかそのイメージを取り払わなくてはならない。ユーザー数が少ないのにソフトが売れるというのがこれまでX68000ユーザーの自慢だったはずだ。

こうなったら，これまで以上にソフトの買い支えをするしかない。そして二度とこんな騒ぎが起きないよう，ソフトを買ったら必ずユーザーカードを出して意見をいおう。本当に面白くなかったら「もう買わない」のひと言だってかまわない。それらがきちっと伝われば，次回作が売れなくて違法コピーのせいにされるなんてこともないはずだ。技術力を棚に上げてX68000は処理速度がどうしたとかいわれることもなくなるはずだ。そして，きちっとしたマーケティングに裏付けされた適正な数のクオリティの高い，そしてX68000ユーザーの指向性にあったゲームソフトがしのぎを削る世界がやってくるだろう。 (T)

Oh!Xでは曖昧な情報源に基づいた読者の声をむやみに載せないよう心掛けています。怪しい事情通の意見ほど，危険な波及効果を生みやすいからです。今回は編集部の考えを本文でまとめてあるので，それを前提に，一連の情報の被害にあった方の声をいくつか紹介しましょう。ここではあえて匿名にさせていただきます。

●最近，X68000にコピーユーザーが増えているらしい。いままでX68000はコピーユーザーが少ないと思っていたのに。なんか，とてもくやしい。 Y.K.(17)滋賀県

●X68000ユーザーの半分がコピーユーザーだと他誌で書かれていた。これを見て僕はちょっとショックだった。X68000ユーザーのなかにコピーユーザーがこんなにいるとは思わなかった。コピーしているやつに限って「このごろいいソフトが出ないなー」と間抜けなことをいうらしい。自分で自分の首を締めているのに。 T.K.(17)神奈川県

●X68000ユーザーはパワーユーザーが多いからかどうか知りませんが，ソフトをコピーする人が多いらしく，ソフトハウスの方々が多額の費用をかけてソフトを開発してもあまり売れないらしいので，場合によってはX68000用のソフト開発を打ち切るという警告を出しています。このままだと，X68000用の新作ソフトが0になってしまう。 Y.W.(19)三重県

●エグザクトの「アクアレス」のデモソフトで知ったのですが，このところソフトの不正コピーがはびこっているようですね。もし，こんなことでX68000が廃れたとしたらと，とても心配です。コンピュータはソフトがなければただの箱なのですから。 T.M.(23)福島県

*

情報の曖昧さに疑問を抱いた方も多いようです。状況を自分なりに判断している方の意見をいくつかピックアップしてみましょう。

●最近のX68000バッシングはヒドイ。自分もプログラムを作る者として，その苦労はよくわかります。しかし，ここ数カ月のソフトメーカーの動きには許せないところがあります。SP○やZO○Mなどは買わなければもう作るのをやめるといいますが，コピーしている者にその声は届いておらず，圧力は買っている者にかかると思います。 村瀬 浩則(21)岐阜県

●最近，ソフトハウスからユーザーへ脅迫まがいのことがいわれているように思える。ZOOMに至っては，売れなきゃ作らないとまでいっているように感じる。SPSも某誌某コーナーでいっていたらしいが，正直，SPSの作ったゲームで私が欲しいと思ったゲームはない。それとも，X68000ゲームユーザーは欲しくないソフトまで買わないといけないのだろうか(私はしょうがないから，いらない生中継68を買った)。 井上 和也(22)福岡県

●Oh!Xを除くほとんどすべての雑誌にコピーユーザーのことが書かれていました。でも，SPSは黙っておくべきだったと思う。 川村 洋志(18)大阪府

●最近，少々ソフトメーカーに対して苛立ちを感じます。違法コピーなどといってソフトの定価がどんどん上がってきています。たとえ月に2万3万円とお金をかけても2～3本しか買えない。これでは売れないのは当たり前です。 久松 愛治(22)東京都

●最近，X68000ユーザーの間で最もよく見かける話題は，なんといっても「ズームの広告」でしょう。最近の68ソフトは売れないらしく，違法コピーばっかりとか。本当にそうなのかなぁと思っています。ちょっとソフト会社は勘違いをしているのでは？ 10万人のユーザーのうち1万本も売れたら大ヒットであるはず。X68000はゲームが目的で買った人も多かったろう。しかし，10万人のユーザーがすべてゲームが目的で買った人ばかりではない。山下章氏はコピー率が最も高いマシンとかいっているけど，そんなの98に決まっているでしょう。だいたい「パロディウスだ！」がいくら良質で移植度バッチリでも，とても万人に勧められるソフトとは思えない。「ファランクス」だってマニアにしかうけないソフトだといいたい。SPSでさえX68000から足を洗う考えがあるらしい。あんまり魅力のないアーケードゲームを移植していたって売れるわけはないのに。すべて違法コピーのせいにするのはやめてほしい！ 真面目に買っている人がかわいそうだし，離れていってしまうでしょう。 功刀 和久(22)埼玉県

# 愛読者プレゼント

PRESENT

リバーヒルソフト ☎092(771)3217

## 1 黄金の羅針盤

X68000用 5"2HD版3枚組

9,800円(税別) 3名

藤堂龍之介探偵日記シリーズ第2弾。時は大正，豪華客船"翔洋丸"で起こった事件を解決するのが目的だ。

ズーム ☎011(613)0191

## 2 ファランクス

X68000用 5"2HD版3枚組

8,800円(税別)

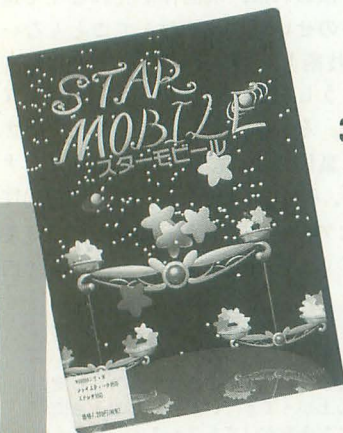
3名

いわずと知れたズームの最新作。横スクロールのシューティングゲームだ。ハードディスクにインストールできるのもうれしい。

M.N.M software ☎0423(60)3084

## 3 スターモビール

X68000用 5"2HD版 7,200円(税別)

3名

上から降ってくる重さの違う星を，天秤にバランスよくのせていくゲーム。パズルはお得意のM.N.Mだけになかなか楽しめる。

### プレゼントの応募方法

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1991年10月18日の到着分までとします。当選者の発表は1991年12月号で行います。

システムソフト ☎092(752)5262

## 4 ALL THAT RPG

10名

システムソフトが発行している，いわば会報ですね。今回は同社のRPGが特集。オールカラー36ページ，非売品。

## 5 清涼飲料水セット 1名

おとどしのモニタの小笠原さんからの提供品。TETSUのアプリコット，アップル，プルーン，グレープフルーツの4風味をセットで。

### 8月号プレゼント当選者

1スコルピウス（愛知県）増田雅光（福岡県）仁宗大輔（佐賀県）北浜晶一 2サブナック（東京都）糸井豊樹（大阪府）若林亮（熊本県）服部直幸 3シューティング68K（東京都）滝澤寿章（栃木県）戸辺靖（愛知県）林知親 4TESORITO（埼玉県）内橋正博（静岡県）勝又裕史（愛知県）白川雄資（大阪府）新子弘康（福岡県）浜地啓ほか5名 5MAXISバッジ（秋田県）清野一男（神奈川県）原田秀孝（群馬県）久保田智久（岐阜県）松原史典（広島県）大岡靖嗣ほか5名 （敬称略）

以上の方々が当選しました。おめでとうございます。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。また，雑誌公正競争規約の定めにより，このプレゼントに当選された方は，この号の他の懸賞には当選できない場合がありますのでご了承ください。

## X－OVER・NIGHT
（クロスオーバーナイト）

### ［第16話］

# 買い換え

TAKAHARA HIDEKI 高原 秀己

最近パソコンを使う友人と話をすると，世代が変わっていることがしみじみわかる。

「W 3がどーのこーの」

「W 3って，ワンダースリーじゃないよな」

「違うってば，WINDOWS 3のことだよ」

「ああ……」

彼は決してフリークでもなければ，オタクでもない。そんな彼をして，最近テクノロジーの波にずるずると飲み込まれてしまっているのだ。

WINDOWSといえば，ぼくがちょうどパソコンの取材をしていた頃の末期に，ようやく「2.0」が搭載され始めた。そう，幻のパソコン規格(おっと，まだあるね，失礼)「AX」が誕生して，三洋電機のマシンにはいち早くWINDOWS2.0が標準装備された。NECも別売で用意したとか，某社がどうするとか，ようやくマック以外の普通のパソコンでも窓表示OS (最近はGUIなどとシャレた呼び方をするらしいが，一般の人には不親切極まりない) が話題になりつつあった。

で，そのうち「3.0」が出てきて，なんでも，「2.0」では従来のアプリケーションソフトは動かなかったけれども，「3.0」では動くというのが売り文句だった。確かPC-9801シリーズに出てきたのは昨年のことだったと記憶しているが。

実際，WINDOWS3.0になると，パソコン通信をしながら，1-2-3と最新ゲームとを併用して，通信から流れてきたデータをひょいと1-2-3のシートに流し込んだりできるはずなのである。事実，こんなイメージに近いことができる場合が多いそうである。

これは2.0ではできなかっただけに，いよいよ「理想のマルチウィンドウ環境」がマック以外のパソコンに登場する日も近いようだと思ったが，ふと考えてみると，それほどいいものなら爆発的に流行するはずだ。してないのはどうしてなのだろうか？

「各タスクの動きが遅いんですよ，それとメモリ3.6Mバイト以上でハードディスクを搭載した高速システムでないとWINDOWSって動かないし」

つまりここで判明するのだが，WINDOWSの実用上の問題点は，AXパソコンのデモで見た時点とさして変わっていないようなのである。つまり，ハードとして相当速いマシンでないと，ちゃんと動いてくれないということ。3.0になると，この要求はますます厳しくなってくるはず。

ここでようやく，先月号で約束した話題へと移ることになる。つまり，ぼくがマシンの買い換えを考えているということだ。なぜ買い換える必要があるのだろう？　答はただひとつ，ぼくが「VM」という文字がデカデカとCPUボックスにプリントされている某機種をいまだに使っているからである。Oh!Xなので機種名は露骨には書かないが。

このマシン，すでに発売から6年近くもたっている。日進月歩，3カ月現場を離れていると，さっぱりわけがわからなくなるほど目まぐるしいパソコンの世界にあっては，「VM」などアンティークもいいところである。

パソコンを使っていると人に自慢しないまでも標榜するならば，標準的な機種における主要テクノロジーは常に使える環境でなければならない。なにかで制限が出てしまうということは，その時点で最低3年間はドロップアウトしてしまうことを意味する。というのは，次に買い換えをする場合はその不足した機能を補って余りある機能を備えた機種が出てくるのを待たざるをえないからだ。目まぐるしい世界ではあるが，主要テクノロジーともなると，サイクルはだいたい3年と見て間違いない。その3年間，新技術から取り残されれば致命傷になりかねない。3年間はそれほど長い。

幸い，この3年間は，ラップトップ戦争ばかりがモテはやされていたので，卓上機種の変遷はさほどなかった。実際には286マシンから386マシンへの変遷が急激に進んでいたわけであるが，シングルタスクにおける現在の使い方を見るかぎり，スピード面以外の問題はさほど表面化しなかったということだ。

さて，諸々の情勢を見るかぎり，買い換えるならば，もう386マシンにせざるをえない。286では今から買うには明らかに役不足。メモリはそのWINDOWS3.0などが問題なく使えるように4Mバイトは積んでいる必要があるだろう。幸い386チップセットもダイナミックRAMもごくごく安い部品と化している。

ただ，問題が2つある。ひとつは「そうはいっても安くはない買い物になる」ということ。もうひとつは「別の機種もいいなあ」という邪心（!?）である。やはり50万円という投資には二の足を踏んでしまう。ぼくだって，アテはないものの結婚資金とそれにともなう引っ越しなんぞということも，一応は考えている普通の人なのである。あとのほうの問題はさんざん某社マシンでいろんな目に遭遇したので，結局は“日本標準パソコン”の枠から出ないように，という自己規制をすることになるのだろう。TRONマシンでも出ていれば考えたはずだったのだが。

ただ，今使っているマシンも実用上はなんら問題なく動いている。それどころか，最新機種を自慢している人よりも，上手に文書作成，通信，表計算，ゲームを交互に使っているとも思う。だけれども，そういって自慢することは，

「ワシはこの車で十分じゃあ。技術は腕でカバーしちゃるけん」

とポンコツ車に固執するおじさんみたいで，不安なのである。

ガンダムF91を見ると，ガンダム初代機は博物館に収められていた。そういえば，その初代機に乗っていたアムロ・レイがなにかの機械を見てつぶやいていたっけ。

「こんな古い機械(モノ)を……」

KYOKO IN ANOTHER CG WORLD
デート♡
だから
るすばん
たのむわね
おねーちゃん
あそぼー
ホンマかいな…
ウィ〜ン
カタカタ
つる〜
ちわーー
MADE IN JAPAN
ヨヨ…
かめ〇〇

コンピュータは 大切に…

というわけで
今回のCGデータは,
物体数　179
光源　　3
1280×1024 PIXEL
1670万色フルカラーを
4×5ポジで出力
使用ソフトは
C-TRACE, サイクロン
Z's TRIPHONY
雲は, 完全透明体に
ペイントソフトで作った
データを貼り付けて作成。

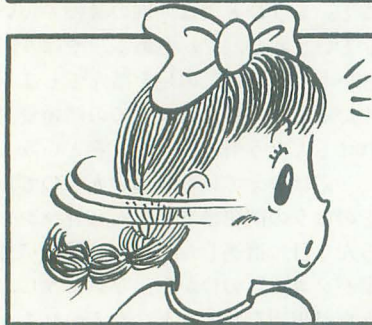
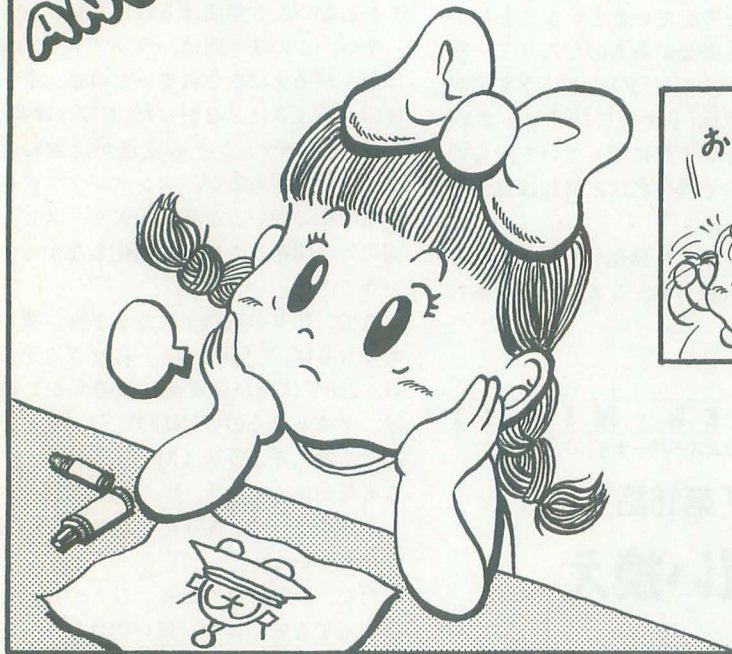
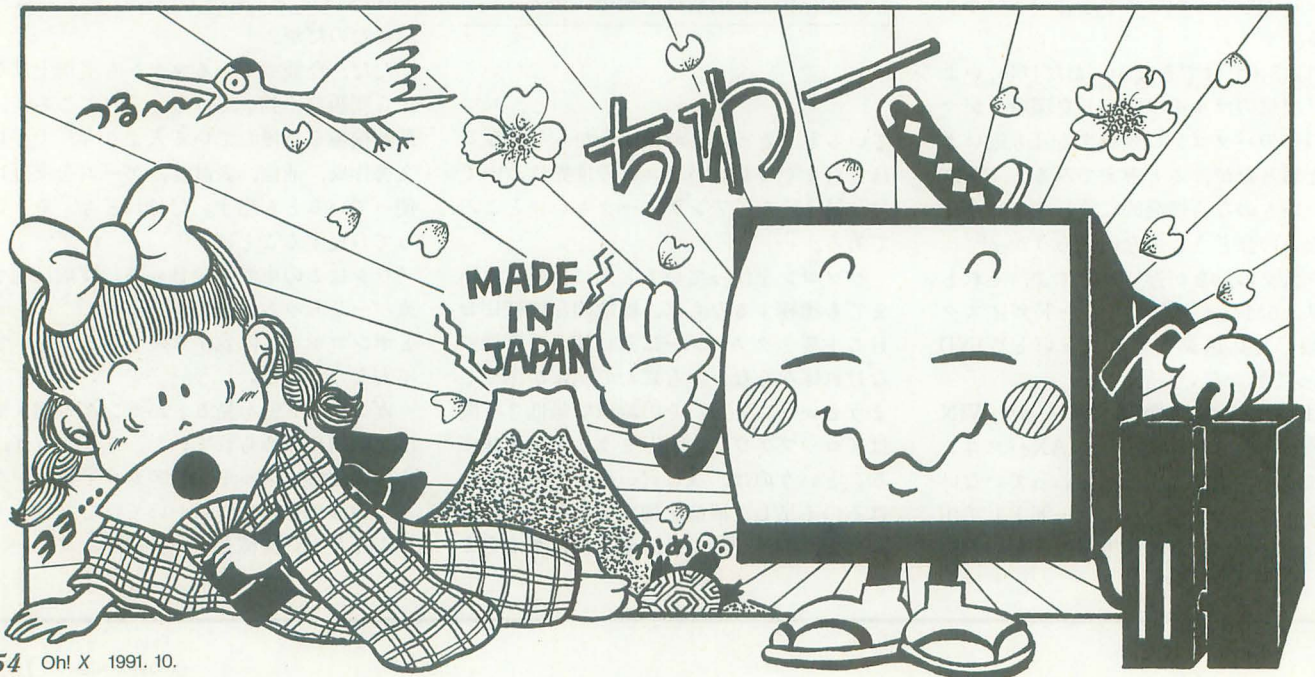
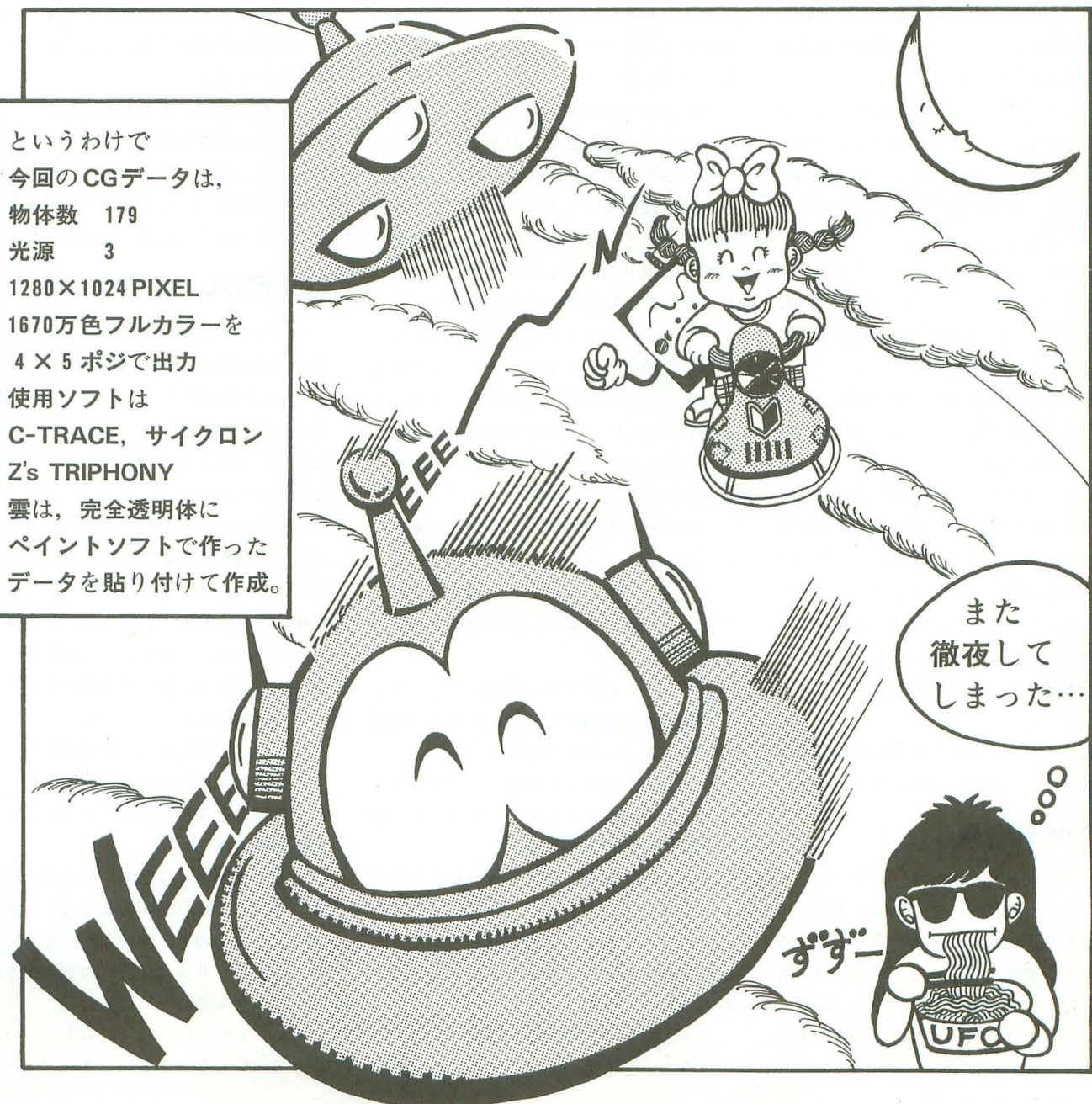
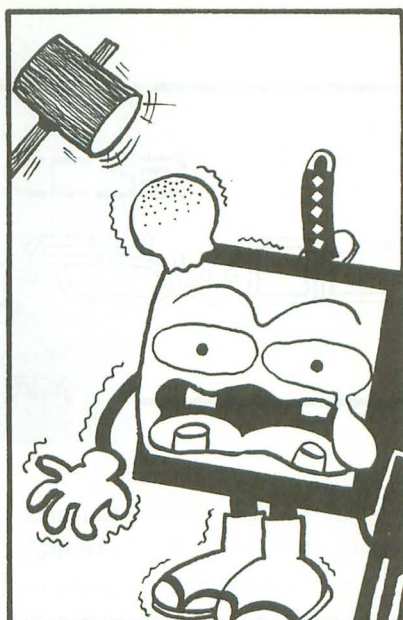
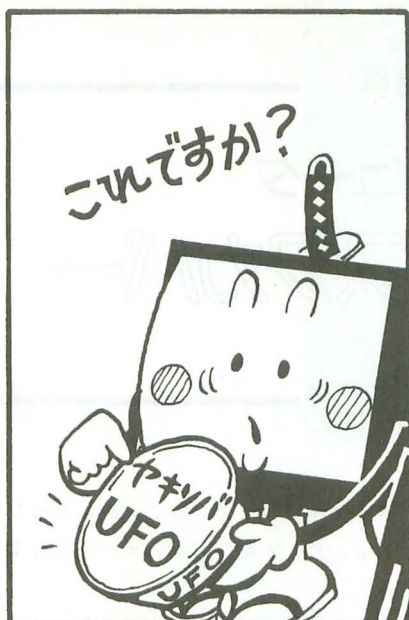
WEEE

WEEE

また
徹夜して
しまった…

ずずー

UFO

第63回

# 猫とコンピュータ
# コロッとディスクカバー

Takazawa Kyoko
高沢 恭子

今月のお話は先月からのつづきです。フロッピーディスク装置の不良原因を知るため，カバーをはずしてみたキョウコさん。ホッとしたのも束の間，また別のフロッピーディスク装置がおかしくなって……。

ブロックゲーム「Yet Another Column」(X68000用ソフト)の得点は，その後伸びていない。いくら奮闘しても3万点がせいぜいで，4万点に届くことはまずなさそうだ。トオルの5万点なんて驚異だ。これが実力というもので，ファイトだけではこの壁は越えられない。

なのについ誘われて手を出してしまうのは，もしやというラッキーなツキをさずかるのが目当てで，それでとてつもない高得点をあげて，自分の運のよさを手柄のひとつにしてみたいからなのか。

ところがこれが，1ゲームを終えるのに10分ほどかかるので，うっかり手をつけたために泥沼に引き込まれて，1時間も遊んでしまうなんてこともある。

どうせやるなら運だめしだけでなく，実力のワクを破る努力をしたらよいのだけれど，これがむずかしい。

まず自分のクセや欠点をみつけて分析する。ここまではわりあいたやすい。ESCキーを使って画面をストップさせ，いくつかのケースを研究することもできる。でもつぎにそれらを改善しなければならない。これは理論ではなくワザをみがくことだから，なかなかきびしい。努力をしても向上するとはかぎらない。

それでもなお，自分の強運に夢をかけるかぎり，ゲームの魔力は消えてくれない。

## ヘッドの脱帽

さて，前回からのつづき。フロッピーディスク装置(FDD)にディスクを差し込もうとしたら，何かがつかえて奥まで入らなくなった。突然のできごとで，あまりにもふしぎだ。いったいどうしたというのだろう。やっぱりこれは中を開いてみなければ気がすまなかった。

メーカーは禁じていることかもしれないけれど，夫と2人でFDDのカバーをはずしてみた。

あたかもS市の家に新しくOAの設備を入れる計画をしているさなか，はたらきつづけてくれたマシンが，何かを訴えるようなトラブルだ。

ボディの左右にある大きなネジをはずしてカバー(外箱)を取りのぞくと，ディスクをおさめるボックス部分があらわれた。

上下2段になったボックス部分はフラットケーブルでつながれており，コネクタをはずすと取り出すことができる。さて，つかえていた上段のボックス内には……。

「ハハ，こりゃひどい」「なにこれ？」

薄い金属でできた小さなハコのようなものが，コロンところがっている。タテ，ヨコ約1.5センチ，高さ9ミリほどのものだ。なんとなくゼリーや寒天を冷やして固める容器を思い出した。

「磁気ヘッドのカバーがはずれて落ちたんだね」

ディスクの上で針のような首をのばしてデータをさがしまわる，あの磁気ヘッドの帽子だった。

「でも，これどうやってついてたの？」

「接着剤か両面テープのようなもので貼りつけてあったみたいだね」

貼り合わされていた部分のポリウレタンらしいものがベトベトになったうえに，ボロボロにすりへっている。

「こんなところに落ちちゃうなんてひどいと思うけど」

「これももう5年くらい使ったかなぁ，つくるほうもここまでは考えてなかったんだろうね」

「もとどおりになる？」

「うん，でもカバーはなくてもいいんじゃないかな」

ためしにカバーのないまま動かしてみたら，以前と変わりない。上が落ちれば，ほどなく下も落ちてしまうだろうというので，下の段のカバーも，はずしてしまった。

「なにかぐあいの悪いことでもおこらないといいけど」

「そのときは，また考えようよ」

ヘッドの重さを調節していたものなのか，ほかからの磁気をさけていたのか，このカバーの役割は気になるところだけれど，いまのところなんの問題もなさそうだ。

## いそがしいからヒマなもの

カバーといえば，TK-80はもちろん，初期のころのPCの各機種(PC-8001/8801)やシャープのMZのシリーズ(MZ-2000/1500)，富士通のFM-8，東芝のパソピア7などに，ことごとく布カバーを手づくりしてかけていた。

棚に並んだマシンがホコリにならないようにと，夫から依頼されたのが始まりだったが，1つ，2つとつくるうち，だんだん熱中するようになった。まだ，私自身があまりパソコンにふれていない時代で，それを使うことより，整理や収納がうまくいくことのほうがたいせつだった。

1台ごとにマシンの寸法をはかり，型紙もつくった。生地はもちろんお揃い。仕上げに機種名をフェルト布でアップリケするという熱心さ。

いくつか手がけるうち，だんだん要領もよくなって，MZ-2000のようにモニタと一体になったものも，なかなかの出来ばえに見えた。ホコリよけの目的は忘れ去り，やみつきの趣味のようになった。

古い「なぞなぞ」で，「いるときにいらないもの，いらないときにいるもの」という

のがあって，おフロのフタや，万年筆のキャップが答えになった。カバーやキャップの役割は，やはり皮肉なものだと思う。

私がいくつかこしらえたカバーも，当然マシンを使っているときは不要で，マシンが休んでいるときに必要だった。そして，活躍するマシンほど，カバーは無用に近かった。

カラーアングル（塗装した鉄枠）でこしらえた棚には，はじめのうちは歴代のマシンが陳列されていたものだが，だんだん込み合ってくると，使われないマシンは押入れにしまわれて，カバーは力作にもかかわらず日かげのものになっていった。

一方，日夜はたらいているマシンのカバーも，しごとのチャンスはなかった。骨なしのヌケガラは，かたわらにみすぼらしく丸められるうち，いつかジャマものとなって，捨てられる日を迎えてしまった。

いまではもう，カバーをこしらえる気はけっしておこらない。マシンにホコリはタブーだけれど，やっぱり，あの冷たさのある堅い肌が整列しているのが，いちばん美しく感じられてしかたがないからだ。

それにしても，ふと目をやると，叩いているキーボードのすき間に見えるたくさんのホコリはひどい。

ふだんはブラインドタッチなので，キーボードをみつめることがあまりないが，つくづくながめてみたら，いろいろなものが落ちている。

小さな細いハリガネ，繊維のようなホコリ，なにやら光る破片。穴あけパンチで打ち抜かれた丸い紙片。すこし拾い出す努力をしてみたが，10分の1も取れない。キーボードにはカバーをしなけりゃかわいそうだけれど，役に立つパソコンほど，カバーはじゃまになる。

## もうひとつのジョウダン

使いなれたFDDの「上段のフシギ」は，とりあえず解決できた。

ところが数日あとに，舞台を変えて，あらたな「FDD上段の怪」が発生した。

S市の家には，やはり32ビットのマシン，PC-9801 DS2を入れることになり，知人のハマノさんの店から購入することに決めた。

ハマノさんとは，まだ知り合ってから日が浅い。パソコンショップを開くために準備をすすめていたハマノさんが，パソコン誌で見た夫のところに，店の紹介をかねて電話をかけてきたのが始まりだった。住まいが同じ町内だったこともある。

明るい表情，かざり気のない話しぶり，パソコンもメカも，心から好きといった若いハマノさんだ。

そしてパソコンショップは，この，地下鉄T線N駅から2つ先の，U駅の前に開店した。ハードもソフトも売り，しかもパソコン相談にはなんでも応じるというのが呼びものの店だそうだ。

新しいマシンは，どこから買うか。価格だけ考えたら，独特の販売法で知られるアキバの「S」の安さがいちばんだけれど，パソコンが大好きで知識も深い人が，1つひとつ，客と語り合いながら扱う商品には，なにか別の満足感を持たせてくれるものがある。信頼や安心の重みだろうか。

PC-9801 DS2は，ハマノさんのショップからS市の家に，宅配便ですみやかに届けられた。

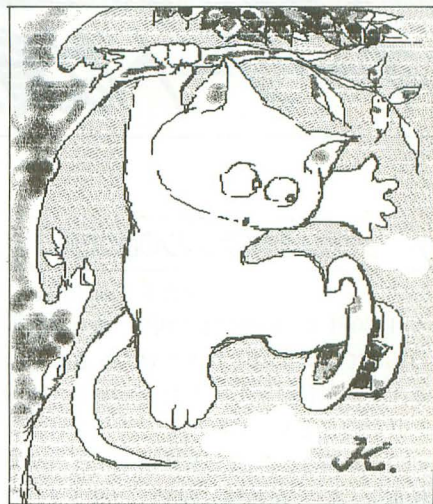
その新しいマシンが，どうしたことか使用後3，4日で，5インチのFDDの上段に異変を生じたのだ。

システムディスクを入れても，ランプがついてカチン，カチンとアクセスする，あのFDD特有のスタートの動きを見せてくれない。なんだかフロッピーディスクが入っていないというような反応なのだ。

ドライブBにディスクを入れると，問題なくアクセスする。つまり，ドライブAだけは，ディスクが無視されたかたちになり，どうやらフロッピーディスクが入ったことを感知するシステムが故障してしまったらしいのだ。

まるで東京のマシンと呼応するような，FDD上段のトラブルだ。

あれこれ考えたり，3.5インチのドライブをつなげて代用させたりしてみたが，購入したばかりのことでもあるし，やはりハマノさんに相談してみようということになった。

ようすを話してみれば，何か解決のヒントを得られるかもしれない。最悪の場合はFDDは東京に返送ということになるだろう。そんなつもりの相談だったのだが，なんとハマノさんの返事は「修理に行きましょう」だった。

illustration：Kyoko Takazawa

「責任のあることですから」と，閉店後の午後8時すぎに車で東京を出発。すこし道に迷ったそうで，11時半，S市の家に到着した。そして，びっくり。奥様もいっしょだった。

奥様のリョウコさんは，神経科のお医者さま。ハマノさんのお店の隣に，やはり医院を開業されたばかりだ。こちらは恐縮するけれど，仲よしのご夫妻には，しごとついでの深夜のドライブかもしれない。

とはいっても，それぞれ1日のつとめを終えたあとなのに，遠方もいとわず，利益もガソリン代も無視した大サービスにつとめてくれるなんて，大感激だ。

ハマノさんの診察の結果，Aドライブは，この場で丸ごと交換しようということになった。新しいマシンでも，たまにこういうことはあるそうだ。

FDDが解体されて，東京の家のFDDと同じ様相になる。

「PC-9801VM2のほうは，ヘッドカバーが落ちましてね」と夫がいうと，

「落ちましたか。最近ではディスクドライブも改良されて，ヘッドカバーのついたものは，もうなくなりましたね」とのことだった。改良前の機種のカバーをはずしてしまったら，どうなるのか，すこしばかり不安になってきた。

リョウコさんも楽しそうにお手伝いをしてくださり，FDDの上段は無事に交換が終わった。午前1時すぎ，「これは最長距離の出張修理でした」と笑顔を残して，ご夫妻は帰路についた。

今月の教訓。FDDは特に上の段をたいせつにしよう。

**第52回　知能機械概論——お茶目な計算機たち——**

# キーワードは貧乏

## 今も思い出すホテルマンの困った顔

プリンスホテルの堂々たる正面玄関に，1日400円（1時間でも300円）のレンタル自転車でセコセコと乗りつけたときには，さすがに気後れがしました。しかし，ホテルマンのひとりに自転車をどこにおけばよいのかを聞き，平然とその指示に従って自転車を（車の）駐車場に移動させました（相手は数秒間表情を曇らせましたが）。

気弱そうなホテルマンは不安に思ったのか，駐車場のすみに自転車を置く瞬間を確認するまで早足でずっとついてきました。僕は自転車のカギを抜き，「ここでいいですか？」と確認したあとに，ごく自然にこういってしまったのです。

「カギは預けますか？」

ホテルマンの顔が見る見るうちに困った表情になってきたので，彼が言葉を発する前に，「あ，自分で持っておきます。」といわざるをえませんでした。

並列処理に関するワークショップが，7月半ばに北海道の函館大沼プリンスホテルで開かれたので参加してきました。案内状の中にはプリンスホテルに宿泊する人に対する案内も含まれていました。それによると，シングル1泊3万円也！　ということで，びっくりすると同時に（別の宿に泊まればいい話であることも忘れて），学会というところは金持ち以外は門前払いなのか？という気持ちにもなりました（実際，それは一面の真理です。論文を載せてもらおうとすると，10万や20万は飛ぶのです，原稿料をもらうのではなくて払うのです）。

でも，そんな心配は無用でした。近所に公務員共済組合の安い旅館があり，早速予約しました。ところが現地に着いて驚きました。旅館からホテルまでの交通手段がなく，歩いたら1時間以上かかりそうな距離なのです。そこで，冒頭のようにレンタル自転車でホテルへ数日間通ったのでした。

さて今回は，国家的な規模になるとゴルビーをも失脚させるという「貧乏」について，この暑いなか考えてみました。

## 計算機科学研究と貧乏

たしかに現在，僕は経済的に裕福であるとはいえないでしょう。はっきりいって貧乏です。この自信に満ち溢れた宣言は，単にプライベートにおいてだけでなく，研究環境という点においてもいえます。

一般に大学というところは，「所属組織の利益を優先することによって生ずる制限」と「貧乏」の取り引きをすでに終えている場所なのであり，いまさらそれに対して不平をいうつもりは毛頭ありません。それどころか，逆に，研究そのものと貧乏との間にはいかなる関係があるかなどというあまりにも唐突な問いかけをして，それに対して，実はそれこそ本質的なのだとさえいおうと思うのです。

計算機に関する研究と限定しないでも，最終的には何かものを作る学問（これは「工学」という言葉の定義なのでしょう）においては，少なくとも両者の間に密接な関係が必要とされるということには間違いないと思われます。そして，この貧乏との関連性こそが，工学と他の学問を分ける重要な目安となっていると思われます。

そもそも，値段あるいは価値という概念が生ずるのは資源が有限であるからといえます。有限な資源を使っていかに効率よくものを作るかということは，純粋に学問的な問題だと考えられます。資源の有限性を表すひとつの重要な尺度が値段であり，それを低くしようという意思が比較的強く働く状態を貧乏と呼ぶといえましょう。

計算機科学という学問分野だけに着目した場合にいえることは，ソフトウェアや計算理論などではものの値段というものが直接絡んでくることはそう多くなく，逆にハードウェアでは直接関係することが多いということがいえます。

もし，ハードウェアの研究において，最高の性能を持つ部品を無数に使えるという前提があるならば，現在なされている多くの研究は無意味なことになってしまうといえるでしょう。ここで，注意しなければならないことをひとついっておかねばなりません。ものの値段の絡んだ学問はなかなか普遍的な，つまり未来にわたっても通用する話とはなりにくいということです。なぜならば，ものの値段は技術の進歩でダイナミックに変化するからです。

## 研究と貧乏に関するケーススタディ

研究と貧乏の関係について抽象的に述べてもいまいち筋の通った話になりにくいので，具体的に例を挙げてみましょう。

**ケース1**

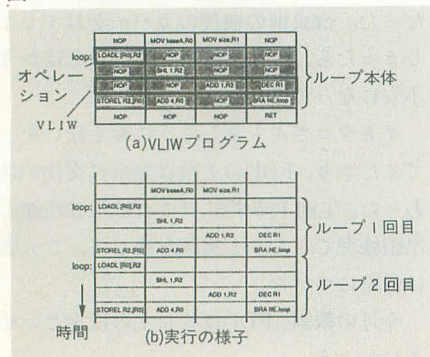
たとえば，メモリの階層構造（レジスタ—キャッシュ—主記憶—2次記憶）に関する研究も，それぞれの記憶素子が1ビットあたりいくらかという値段が前提として成り立っている当たり前の話といえます。値段を考えないのならばいちばん速いメモリだけを大量に接続すればよいのですから。

**ケース2**

工学において，値段というものは純粋な学問的なレベルに入るべきであるという主張は，従来はあまり受け入れられないものでした。たとえば，計算機アーキテクチャの教科書でも具体的な値段が載るようなものはまれでした。しかし，最近ではそうでもなくなる傾向にあるようです。その賞賛すべき例が本連載の前回で取り上げた，「今世紀最後の計算機アーキテクチャの教科書」[1]です。その中では代表的な計算機をいくつか取り上げ，具体的にどのようにして，我々が買う計算機の値段が決まるかということが部品の値段から小売店での割り引き率まで考慮して書かれています。

**ケース3**

函館のワークショップでも，ある並列計算機を使ったという発表で，そのアーキテクチャの説明をするときに，さかんに予算の都合でこうなったと繰り返していた人がいました。さらに，発表後の質問で学術的な意味合いを問い正したのに対しても，お金がないからこうなったなどと繰り返していたので，質問者も腰が砕けてしまった様子でした。発表者にもいろいろな事情があ

**図1　VLIWプログラム**

るのではないかと思いますが，研究を外から圧迫する具体的なものとしての貧乏はやはり寂しいものといえましょう。

**ケース4**

ちょっと前の学会（全国大会）で，大学院生がずいぶんと古い計算機の上にアセンブラを作ったという発表をしました。聴衆にもどこに発表の意味があるのかわからない様子で，質問がひとつも出ずシーンとしていました。そこで座長（司会）の先生はひと言だけこういいました。「先生に頼んでもっといい計算機を買ってもらいなさい」

## VLIWと貧乏の融合

VLIW（Very Long Instruction Word）というアーキテクチャがあります。そのアーキテクチャでは従来のひとつの命令を複数個連結してひとつの命令にします。そして，まるでふつうの逐次型プロセッサのように，機械語命令をひとつずつ持ってきては順に実行するのですが，実はその命令は複数の命令から構成されているので，プロセッサ内ではそれらの命令が並列に実行されるのです。もちろんプロセッサ内には並列に動作可能な機能ユニットを複数個用意しておく必要があります。

VLIWは比較的容易に並列計算を実現できるという点で大きな魅力がありますが，当然短所もあります。そのひとつが，機械語1命令のビット数がきわめて大きくなってしまうので，プログラムメモリ量，プログラムメモリとプロセッサ間のバンド幅，各機能ユニットまでのプロセッサ内のハードウェア量が大きくなることです。

それを改良しようということがひとつの大きな柱となっているのが，今回の北海道で僕が発表した「実行遅延に基づく再構成VLIW型計算機」[(2)]です。VLIWと貧乏という概念の融合といってもいいでしょう（まさか！）。内容は比較的明瞭なので，ちょっとだけ紹介してみることにします。

まず，図1に従来のVLIWマシンのイメージを示しておきます。(a)に示すのが，VLIWマシン用のプログラム（のループ部分）です。4つの並列に動作する機能ユニットを想定しており，横方向の4つの命令（オペレーションと呼びます）が1命令（VLIWと呼びます）を構成しています。このプログラムを実行したときのタイミングを表したのが(b)です。ループ1回あたり4ステップかかっているのがわかります。

さて，肝心の「実行遅延に基づく再構成VLIW型計算機」ではオペレーションごとに，指定したサイクル数だけ実行を遅らせることができるのです。図2にVLIWの再構成の概念を示します。図の左側(a)にはプログラム（メモリからプロセッサに持ってくるタイミング）を示しています。記号をつけていないオペレーションはNOPです。ここではすでにBとFという2つの命令に実行遅延のステップ数（この場合両方1）がコンパイラによってつけられています。

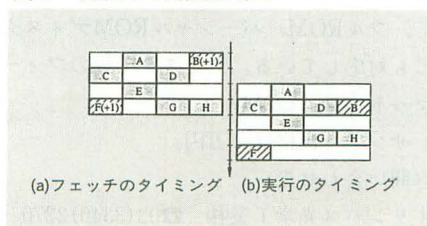
さて(a)に示されるプログラムを実行したときのタイミングを(b)に示します。たしかにBもFも1ステップ，プロセッサ内で遅らされており，プログラムメモリに格納されていたのと違う組み合わせのVLIWとして実行されているのがわかります。

さて，このようなメカニズムを採用すると何がうれしいのかひとつだけ例を挙げましょう。図3(a)にサンプルプログラムを示します。従来のVLIWマシン用のプログラムです。ループ1回の実行にはどうしても3ステップかかります。

(b)に再構成型VLIWマシンのためのプログラムを示します。ひとつのオペレーションにだけ2ステップの実行遅延が指定されています。実行させたタイミングを表したのが(c)ですが，なんとループ1回の実行が2ステップになっています。

ですから，このループ部分についていえば，プログラム量も実行時間も両者とも3分の2に軽減されているということがいえるわけなのです。なにかきつねにつままれた気がする読者の方もいるのでは？

手前ミソながら，ほかにもいろいろあるのですが，その議論をここで展開するのは，やりすぎというものでしょう。

**図2　VLIWの再構成の様子**

## 貧乏暇なし

原稿を書いている横で長良川トライアスロン大会のテレビ中継をやっています。特に見入っているわけではないのですが，トップがやたら入れ替わったり，女子のトップが急に苦しみながら横になったりするので，案外意識が画面にいってしまいます。

番組の最後に，なぜか解説をしていた元ラグビー選手の松尾がいみじくもこんなことをいいました。

「この暑いなか，手足を動かすのは貧乏人だけというのに，よくがんばりましたね」

さすがにこれを聞いたときは，アイスコーヒーを作ってごろごろと15分ほど時間をつぶしてしまいました。ふと，「もしいま一生遊んで暮らせるような大金がころがりこんだとしたら，僕は研究者としての生活を続けるだろうか？」という愚問が浮かんできます。これは難問かもしれません。経済的に余裕ができたら，じっくりと考えることにしましょう（貧乏暇なし）。

**参考文献**

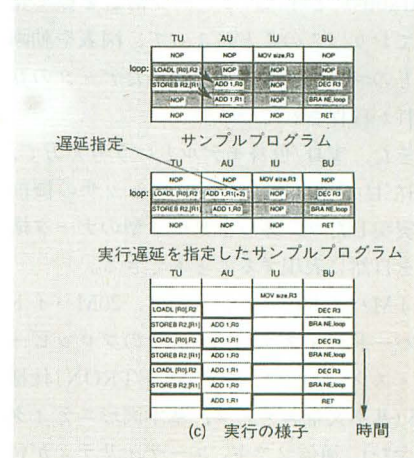
(1) D.A.Patterson and J.L.Hennessy, "Computer Architecture-A Quantitative Approach", Morgan Kaufmann Publishers, 1990

注：本誌7月号でこの本を取り上げ，筆者の名前を公平に扱うように筆者が求めているということをその中で紹介しましたが，その後一部の読者の方々から，あの記事ではHennessyの名前が先にきており，不公平だという親切なおしかりをいただきました。今回はPattersonを先に書きましたので，やっと公平になりました。

(2) 有田，加藤，曽和：実行遅延に基づく再構成VLIW型計算機の基本構成，1991年並列/分散/協調処理に関する「大沼」サマーワークショップ

**図3　ループのオーバラップ実行の例**

# PENGUIN INFORMATION CORNER

## ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

## NEW PRODUCTS

一般向けBTRONパソコン

### 1B/note

パーソナルメディア

1B/NOTE

総合パッケージソフトウェアメーカーのパーソナルメディアから，BTRON仕様パソコン「1B/note」が発売される。BTRON仕様のパソコンが一般向けに発売されるのは今回が初めてのこととなる。

この「1B/note」は，BTRON仕様に準拠したOS「1B」をノートタイプパソコンに実装したもので，GUIを使った基本エディタやパソコン通信の機能がはじめから付属している。

BTRON仕様OSは図表や動画，音声といったマルチメディアの処理を考慮して設計されたOSである。TAD(TRON Application Databus) と呼ばれるデータ構造を標準化しており，このことによって，図表や動画などのマルチメディアを含めたデータの互換性が確保されている。

また，実身/仮身モデルという考え方で，OSだけでアウトラインプロセッサの機能を実現したり，ネットワーク型のデータ構造を自然に表現することもできる。

4Mバイトのメインメモリ，20Mバイトのハードディスク，3.5インチのフロッピーディスク，マウス，モデム，BTRON1仕様OS(基本文章エディタ，基本図形エディタを含む)，通信ソフト，ユーティリティが付属していて，CPUは80386SX (16MHz)。

価格は485,000円だが，9月末までなら398,000円，11月末までなら450,000円と，期間限定の特別価格を設定している。

現在，予約受付中で出荷開始は9月末の予定。初年度に3,000台の販売を予定している。

〈問い合わせ先〉

パーソナルメディア㈱　☎03(5702)0355

業界最高速の3.5インチMO

### MOS300E

オリンパス光学工業

MOS300E

オリンパス光学工業は，現在商品化が発表されているなかでは業界最高速の書き換えが可能な，3.5インチ光磁気ディスクドライブを商品化。コンピュータメーカー，システムインテグレータ，VAR(付加価値再販業者) などに幅広くOEM販売する。

本ドライブは平均アクセスタイム46ms (ミリ秒)以下，回転速度3,600rpm(回転/分)で，768Kバイト/sという連続データ転送速度を実現している。

SCSIコントローラ内蔵で，SCSI-2インタフェイスを採用している。また，ISO規格案準拠の3.5インチ光磁気ディスク，そして，フルROM，パーシャルROMディスクにも対応している。ディスク1枚のフォーマット容量は128Mバイト。

サンプル価格は27万円。

〈問い合わせ先〉

オリンパス光学工業㈱　☎03(3340)2270

薄型でカラフルな3.5インチFD

### ニュー「スリム」シリーズ

富士写真フイルム

富士写真フイルムは，通常の1枚入ケースと同じ大きさのケースに2枚のフロッピーが入る3.5インチフロッピーディスク，「スリム」の新シリーズ，ニュー「スリム」を発売した。

今回のニュー「スリム」シリーズでは，新たに透明ケース入りカラーシェル品 (ブラック，ブルー，グリーン，ピンク) が加わった。全部で2品種10タイプとなったことで，いままでのキャリング性と保管性に，ファイリング性が加わった。

〈問い合わせ先〉

富士写真フイルム㈱　☎03(3406)2111

おしゃべり&ミニ言語

### SPEAK SYSTEM/FUNCTION CALL

サザンエンタープライズ

サザンエンタープライズは，X68000用ソフトウェア「SPEAK SYSTEM」,「FUNCTION CALL」を発売した。

「SPEAK SYSTEM」はかな文字で書かれたファイルを発声させるもので，デバイスドライバとユーティリティツールがセットになっている。またこのシステムで，同社の「DiSS-P」の各データをほかの環境で活用することもできる。

「FUNCTION CALL」はIOCSやDOSな

どのファンクションコールをコマンドラインから実行するためのミニ言語である。

価格はともに2,000円（税別）。

〈問い合わせ先〉

サザンエンタープライズ☎03(3787)3932

F-1オリジナルパッケージ
## 「写ルンですHi/FLASH」
富士写真フイルム

写ルンですFLASH

富士写真フイルムは，現在若者を中心に圧倒的人気を誇るモータースポーツ界最高峰のF-1をデザインした，「写ルンですF-1オリジナルパッケージ」を限定発売している。

この製品は，富士写真フイルムがスポンサーをしているチーム，ジョーダングランプリを「写ルンです Hi/FLASH」にデザインしたもので，価格は通常品と同じく，1,000円と1,800円となっている（ともに税別）。

〈問い合わせ先〉

富士写真フィルム㈱　☎03(3406)2111

# INFORMATION

新たに25種類，全35種類
## 「アスキーネット」サービス拡大
アスキー

アスキーは，パソコン通信サービス「アスキーネット」に新たに25種類のデータベースを追加し，従来から提供している10種類とあわせ，計35種類の情報サービスを提供することになった。

今回から新たに提供されるサービスは，パソコン通信「アスキーネット」から総合データベース「G-Search」を利用できるようになったことにより実現されたもので，音楽や図書目録など５種類の趣味の情報と，企業情報や新聞，通信社の記事情報など20種類のビジネス情報がある。

趣味の情報では，５万件の音楽CD情報とジャズ，ロック，クラシックなど，あらゆるジャンルの音楽情報を収録した「音楽CD総カタログ」，毎月１千件の新譜CD情報が追加される「CD新譜情報」，国内約13万冊の出版物を収録した図書目録情報などがある。

一方，ビジネス情報では，企業情報データベースの「帝国データバンク企業情報」や，「東京商工リサーチ企業情報」，新聞や雑誌などの記事を収録した記事データベース，著書やプロフィールを検索できる23万人の人物，人材情報データベースなどがある。

「アスキーネット」は，いままで力を入れてきた電子掲示板や電子メールなどのコミュニケーションサービスや機能に加え，今回のデータベースやニュースなどの情報サービスの拡大で，幅広い顧客ユーズに対応するようになった。

〈問い合わせ先〉

アスキーネット事務局　☎03(3486)9661

オーディオフェア40周年記念
## デジタルサウンド・コンテスト
日本オーディオ協会

日本オーディオ協会では学校法人尚美学園の特別協力を得て，コンピュータとシンセサイザによる音楽制作コンテスト「デジタルサウンド・コンテスト」を開催する。

募集されるのは「コンピュータミュージックの部」と「キーボード，シンセサイザの部」の２つ。ともに５分以内の音楽作品が対象で，入賞作品は第40回オーディオフェア特別会場にて制作者によって演奏発表されることになる。

応募要領は以下のとおり。

○コンピュータミュージックの部

コンピュータ，またはコンピュータと電子楽器のシステムを使用したオリジナル曲，既成曲。使用機器，およびソフトウェアは自由。

○キーボード，シンセサイザの部

キーボード，シンセサイザなどの電子楽器，機器を効果的に使用したオリジナル曲，既成曲。映像を含めた作品もビデオテープで募集。使用電子楽器，機器，および形態は自由。

○賞

最優秀賞　１作品

優秀賞　各部門より１作品ずつ

佳作　各部門より１作品ずつ

最優秀作品に30万円以上の電子楽器などが贈られるほか，各賞ごとに賞品が用意されている。

○募集期間

10月３日㈭まで

○応募先

〒112　東京都文京区小石川1-1-8
(学)尚美学園「デジタルサウンド・コンテスト」事務局

〈問い合わせ先〉

デジタルサウンド・コンテスト事務局

☎03(3818)7691

## 第１回全日本X68000芸術祭地区予選大会開始
シャープ

第１回X68000芸術祭の予選大会が全国のトップを切って７月21日に高松で，そしてそれに続き，８月４日には北海道で開催された。どちらの大会でも，場内満員となる約220名の来場があり，大変な盛り上がりを見せた。

高松大会では，ミュージックプログラム「X68000な感じ」（岡崎一平），豊富な機能のスプライトエディタ「C_32C.X」（金沢充泰）との激しい票争いの末，バグ退治ゲーム「LOGICRUSH」（鴨居大吾，藤原智行）が大賞受賞。

北海道大会では，SX-WINDOW上でMIDI演奏を行うためのアプリケーションソフト「SX-MEGATONE」（濱田淳一）が来場者の圧倒的な支持を得て，見事大賞に選ばれた。

全国で行われるこの予選大会の大賞受賞作品は，来年４月に開催が予定されている全国大会に各地区代表として出場することになる。

〈問い合わせ先〉

シャープ㈱　☎03(3260)1161，06(621)1221

# FILES Oh!X

このインデックスは，タイトル，注記——著者名，誌名，月号，ページで構成されています。運動会や文化祭などなにかと忙しいこの時期。おいしいものをしっかり食べて英気を養いましょうね。

## 一般

▶特別企画 最新パソコン選びのAtoZ
「○○○をするには△△△を買えばいい」式で，目的を持った方のパソコン選びの参考になる，パソコンと周辺機器の選び方ガイド。——北澤充裕，マイコンBASIC Magazine，9月号，67-78pp.

▶チャレンジ！ ザ・CG
パソコンCGを描くうえで必要なソフトやハードの紹介。色の構造や実際に描くときの手順などを解説。アニメ調の絵を描く際に参考になる。——編集部，POPCOM，9月号，96-105pp.

▶第1回どのハードがエライ？ 選手権
各機種のゲーム処理速度を比較。X68000は遅いそうである。——編集部，POPCOM，9月号，121-125pp.

▶特集!! どーするどーなるパソコンゲーム［移植編］
パソコンゲームの機種間移植がめっきり少なくなってきた。これはいったい何が原因なのだろうか？ ソフトハウスへのインタビューデータなどを基に考えてみる。——編集部，テクノポリス，9月号，127-132pp.

▶アルゴリズムを見切ったぞ!?
データ圧縮の基本アルゴリズムを解説。X68000用にX-BASICのサンプルプログラムも掲載。——おにおん・竹内信和，テクノポリス，9月号，154-158pp.

▶NETWORK CONNECTION
パソコンがなくてもパソコン通信サービスが受けられる，「アスキーQネット」スタートの話題と，PDSデータのやりとりに欠かせないISH形式ファイルについて解説。——編集部，LOGIN，15号，280-281pp.

▶日本パソコン百景
織機とパソコンの関係を探るべく，東京都八王子市の中村織物にでかける。縦糸の上げ下げの情報は，紋紙というパンチカードに記憶され，織機はそれによってパターンを織っていく。さらにコンピュータを使った電子的な制御も始まっているとか。——フデヲシ＆カワラ，ASCII，9月号，206-207pp.

▶失敗しないプリンタ選び'91
プリンタとひと口にいってもバラエティがあり，利用目的と要求されるスペックを知らないと大きな失敗をすることもある。この特集ではプリンタのタイプと特徴，用途別の向き不向き，ベンチマークテストなどで各種プリンタをチェックする。——編集部，ASCII，9月号，210-232pp.

▶パソコンで体験する天文学
今回は連星について種類や形成のしくみを解説し，その運動をシミュレートする試みと，銀河形成爆発説にのっとって宇宙地図をディスプレイ上に再現する試みを行う。——福江純・岡田理佳，ASCII，9月号，274-280pp.

▶Inside of Intelligent Products
今月から始まったコーナー。CPUと複雑なプログラムを搭載した未来的商品のしくみや実現方法を解説する。1回目はGPSを使って瞬時に現在の位置と高度・時刻を表示するSONYのPYXISを扱う。——編集部，ASCII，9月号，297-299pp.

▶バカパパのモノを買い物
キーボードの周辺グッズを買い揃え，便利度などを考える。アームレスト，キーボード立てかけスタンド，原稿ホルダーなど，買うのにふんぎりのいりそうなものが総ざらえ。——バカパパ，ASCII，9月号，332-333pp.

▶MEDIA BREAK SPECIAL
今年の6月，ナムコがイギリスのW.Industries社の開発した業務用ゲーム機「Virtuality 1000SD」を導入し，同社の直営店に設置した。その概要を紹介し，ナムコ担当者の話を聞く。——綾丸，ASCII，9月号，345p.

▶低レベルソフトウエア研究所
低レベルソフトウエア研究所純正の圧縮ユーティリティのお話。圧縮率を1.6倍にする，なんとバッチファイルだ。そのしくみは，見てのお楽しみ。——円弱信夫，ASCII，9月号，380-381pp.

▶ディスクメディア大全集
パソコンの必須の装置のひとつ，外部記憶装置。その歴史を振り返り，現在の大容量化を支える技術を解説する。さらにフロッピーディスクとハードディスクの今後の動向を探り，今後の行方を考える。——山田憲一，マイコン，9月号，111-118pp.

▶MYCOM WATCHING
進学校として全国に名を知られる桐蔭学園では，小学部から高校まで創造性の育成を狙いとしてパソコンをカリキュラムに取り入れている。小学生はもちろん，中・高校の女子生徒にもなかなか好評のよう。——菊地秀一，マイコン，9月号，134-137pp.

▶パソコンと教育
平成4年から学習指導要領によってコンピュータ学習が実施されることになった。情報教育の効果やその可能性，計測システムを使った実験の実際などをレポートする。——編集部，マイコン，9月号，138-174pp.

▶マイコンクリーン大作戦
パソコンをキレイに保つメンテナンスについて考えるシリーズ。今月はホコリとヨゴレ対策について。システムカバー，キーボードカバーなどの製品が紹介されている。——猪野清秀，マイコン，9月号，251p.

▶なんでもQ&A
書院AXを少ないメモリ環境で起動させるためにはどうすればよいか，AX386DでWINDOWS3.0を386エンハンスモードで起動させるにはどうすればよいかなどの質問に答える。——シャープ株式会社，マイコン，9月号，410-411pp.


**参考文献**
I/O 工学社
ASCII アスキー
コンプティーク 角川書店
テクノポリス 徳間書店
POPCOM 小学館
マイコン 電波新聞社
マイコンBASIC Magazine 電波新聞社
LOGIN アスキー


## 新刊書案内

前作『コンピュータ社会が崩壊する日』の続編にあたる本書では，前作ではわかりにくかった点が具体的に語られ，「フォン・ノイマンの呪い」という抽象的な言葉もうまく説明されている。ああ，本当に「呪い」だなあ，という気がしてくる。そして，前作で崩壊させた「ノイマン型社会」を救済するのは「マクルーハン型社会」である，と。

最近マクルーハンを取り上げる動きが活発だ。マクルーハンといえば「メディアはメッセージだ」というひと言が有名だが，コンピュータをメディアとして論じられるようになったことが最も大きな原因だろう。では，メディアとしてのコンピュータが発しているメッセージとは，本書によると「創造せよ，感性を増幅せよ」であるらしい。

タイトルになっている「情報新人類」という名の「マクルーハン型人間」が「ノイマン型人間」のあとを継ぎ，日本に蔓延していた「儒教思想」を崩壊させ，未来を作る。そして，コンピュータが文化にならないと，「ノイマン型社会」の終焉とともに，斜陽化していく。非常に刺激的で面白い本だ。平易で読みやすいので，暇潰しでもいいから，一度手に取ることを勧めたい。(K)

**情報新人類の挑戦　逢沢明著　光文社刊**
**☎03(3942)2241　新書判　232ページ　770円**

書籍の価格は消費税込みです

# MZシリーズ

MZ-1500(BASIC MZ-5Z001)

▶MOGU-ら

家族みんなで楽しもう！ コンピュータもぐらたたきゲーム。——小川幸泰，マイコンBASIC Magazine，9月号，124-125pp.

MZ-2500(BASIC-M25)

▶最終兵器

警備兵の攻撃をよけながら最終兵器をすべて破壊する。デモ・コンストラクション付きパズルゲーム。——謎のパズル大好きおじさん，マイコンBASIC Magazine，9月号，126-127pp.

# X1/turbo/Z

X1シリーズ

▶Heavy cat

吹いてくる風をよけつつバクダンを画面下まで落とす。バクダンが上まで吹き飛ばされるとゲームオーバー。——権田典之，マイコンBASIC Magazine，9月号，153-154pp.

▶KURU KURU 2

画面左上にあるコマを右下まで持っていくパズルゲーム。全200面。——中村理，マイコンBASIC Magazine，9月号，155-157pp.

▶COMMANDER

敵と味方とが入り乱れてリアルタイムに戦うゲーム。5つのフィールドに味方の兵器を配置し，コマンダー・タンクを操作して味方の兵器とともに敵国側を打ち倒すのだ。——Isis，I/O，9月号，146-156pp.

X1＋FM音源ボード（要NEW FM音源ドライバ）

▶ターボ・アウトラン ～Shake the Street～

セガのゲームミュージックプログラム。——伊藤圭一，マイコンBASIC Magazine，9月号，187-189pp.

X1turboシリーズ

▶KNIGHT AND DRAGON

ガノス王国を支配したドラゴンと魔物たち。剣士ジョンはガノス王国を救うことができるか？ フィールド型ファンタジーRPG。——石井一鑑，マイコンBASIC Magazine，9月号，158-160pp.

# X68000

▶X68000芸術祭インフォメーション

オリジナルソフト発表の場でもあるX68000ユーザーのイベント，X68000芸術祭の情報を掲載。全国のトップを切って開催された四国地区大会を速報。——山下章，マイコンBASIC Magazine，9月号，90-92pp.

▶COROL

名前から想像できるとおりの玉ころがしアクションゲーム。海に落ちたり，タイムアウトしてしまうとミス。ミス10回でゲームオーバー。——福田圭介，マイコンBASIC Magazine，9月号，161-162pp.

▶ダウンフォース

車でコースを走ってタイムを出すゲーム。ジョイスティック専用。リタイヤしちゃだめだぞ！——いとうたかゆき，マイコンBASIC Magazine，9月号，163-164pp.

▶DESERTER

トビラに向かって走れ！ カギのかかった扉を開けるために，散らばったカギを拾い集める。邪魔なスライムをやっつけろ！——川名高司，マイコンBASIC Magazine，9月号，165-166pp.

▶MARVEL LAND ～ジェットコースター面BGM～

ナムコのゲームミュージックプログラム。要NAG DRV＋MT-32系MIDI楽器。——吉崎雅敏，マイコンBASIC Magazine，9月号，190-191pp.

▶SOFT EXPRESS

新着ゲームのイースやアクアレス，ドラッケン，キャメルトライなどを紹介。——編集部，コンプティーク，9月号，77-81pp.

▶Software Hot Press

ワイヤーフレームが懐かしい3Dシューティング「3D2」や，深海ワイヤーアクションゲーム「アクアレス」，そのほかアークス・オデッセイ，インペリアル・フォース，キャメルトライを紹介。——編集部，POPCOM，9月号，22-25pp.

▶ミュージック・パビリオン

OPMミュージックデータ。上々颱風の「流れのままに」。——編集部，POPCOM，9月号，167-169pp.

▶GAMING WORLD

発売予定の銀河英雄伝説IIDX＋Kit，3D2，ラストバタリオン，オルテウスII，アクアレス，Noah，ボナンザブラザーズ，フェアリーランドストーリーの制作情報や，新着のアークス・オデッセイを紹介。——編集部，テクノポリス，9月号，14-34pp.

▶NEW SOFT

8月発売予定のロボットアクションゲーム「アクアレス」を紹介。——編集部，LOGIN，15号，17p.

▶X68000新聞

M.N.Mの新作「3D2」やエピック・ソニーが手がけたゲーム「ドラッケン」「ゼノン2」，X68000版「イース」の紹介など。——編集部，LOGIN，15号，262-265pp.

▶NEW SOFT

その名のとおりの3Dシューティング「3D2」や「ゼノン2」「インペリアル・フォース」を紹介。——編集部，LOGIN，16・17号，12-24pp.

▶X68000新聞

ジーザスII，Noah，F15ストライクイーグルII，ドラゴンウォーズ，ロードス島戦記，銀河英雄伝説IIDX＋kitなどの新作ゲーム紹介。——編集部，LOGIN，16・17号，274-277pp.

▶AV STRASSE

AV指向マシンの最新ソフトやPDSを紹介するページ。X68000用のIOCSフォント・200書体，ファイルセレクタsee.x，ESVプレイヤーのSXesv.xなどを掲載。——編集部，ASCII，9月号，321-324pp.

▶FREE SOFTWARE INDEX

主要BBSにここ数ヵ月のうちにアップロードされたオンラインソフトを編集部がチェックし，リストアップしたページ。X68000用PDSも多数紹介されている。——編集部，ASCII，9月号，371-375pp.

▶HOBBY EXPRESS

ボナンザブラザーズ，ボンバーマンが紹介されている。——編集部，マイコン，9月号，355-371pp.

▶なんでもQ＆A

シャープから新発売のディスプレイの特徴，SWITCHコマンドのSCSI_IDの意味，Teleportion PRO-68Kについてなど。——シャープ株式会社液晶映像システム事業部第2商品企画部，マイコン，9月号，408-409pp.

▶Misnon改良版

キータイプ時の間違いをコンピュータ側で訂正・指摘してくれるMisnonに改良を加える。デバッガなどとの相性が悪かったのが，コマンド・ライン上から入力された文字列以外は修正を加えないようにすることで解決された。——まてりある，I/O，9月号，122-128pp.

▶SOFT BOX

SX-WINDOW用グラフィックソフト「Easy Paint」を取り上げる。メモリ消費の環境面から操作性，今後の展望まで解説。——山下利夫，I/O，9月号，197-199pp.

# ポケコン

PC－E500

▶POKA!! POKA!!

タイムが0になるまでに何点稼げるかな？ モグラたたきゲーム。——舘学，マイコンBASIC Magazine，9月号，168p.

▶DANGEON BUSTERD

地球侵略のために地下600メートルのところに基地を作ったアデッサ星人。基地を破壊して，彼らの野望から地球を守るのだ！——中村啓之，マイコンBASIC Magazine，9月号，169-171pp.

▶誌上公開質問状

PC-E500とE550の違いはどこか？ などの質問に解答。——Akiko，マイコンBASIC Magazine，9月号，193p.

**電脳騒乱節　2**

あの「ザ・ベ」のカラーページに連載している「電脳騒乱節」の単行本である。業界内部にいる利点や知識を生かし，パソコン界にまつわるさまざまな人や現象やメディアを切っていく。余談とくだらないギャグ（のようなもの）を交えながらいいたい放題，というのが受けている。この業界ではネタには困らないのだろう。なのに，爽快感を感じないのはなぜか。おそらく，そのメッセージが業界に漬かっているからだと思う。　(K)

**中村光三郎著　技術評論社刊　☎03(3225)3293　A5判　219ページ　980円**

**フリッパーズ・テレビ**

最近，新しい若者のメディア文化を認めようという動きがいたるところで見受けられる。そんな新しい文化をテレビというメディアを通して語ったのが本書である。雑誌などの原稿をまとめたものなのでいささか古いものもあるが，その方向性はいまだに通用する。コンピュータやテクノロジーが従来のメディアに影響を及ぼしたり新しいメディアを作り上げていくという状況の面白さ，一緒に走っていける面白さがポイントだ。　(K)

**稲増龍夫著　筑摩書房刊　☎03(5687)2680　四六判　221ページ　1,450円**

書籍の価格は消費税込みです

# QUESTION and ANSWER

4月号の質問箱を読んで割り込みに興味を持ったのでラスタスクロールをやってみようと思ったのですがうまくいきません。ラスタでもなんでも割り込み先はA1レジスタに指定していますが「垂直帰線期間になるまでラスタごとに割り込む」(こんなことはしませんけど) みたいに連続して割り込むときなどA1レジスタにどんどん番地を入れてしまってもプログラムに狂いは出てこないのでしょうか? さらに変な話ですけど，ラスタとは何者なんですか? 画面に何本のラスタがあるのでしょう? 以上お願いします。

千葉県　川俣　俊弘

ラスタがわからないということですが，ひと言でいえば走査線のことです。たとえば画面モードが768×512なら，512本の水平線を表示することができますが，単純に水平線の1本を1ラスタといいます。つまり“幅768ドットのラスタが512本表示されている”といい換えることができます。

さて実際のラスタスクロールはどのようにしてやるのでしょうか。X68000の画面表示部分(CRTC)はあるアドレスから順にデータを読み込んでそれを色コードだと解釈して映像信号に変換しています。通常は画面モードによって固定されている読み込み位置を書き換えてやれば，スクロールが可能になります。具体的にいうと垂直帰線期間中にCRTCの表示開始位置を指定するレジスタを書き換えるだけです。

実はCRTCは一度に画面全体を表示するわけではありません。ディスプレイの走査線が画面を走るのと同じ速度で，上から下へデータを信号に変換していると考えてもいいでしょう。表示はラスタ単位で行われ，1本のラスタは左から右に走査線を走らせることで表示されます (この期間を水平表示期間という)。このあたりの話は4月号の質問箱でも取り上げたので詳しくは話しませんが，要点は画面表示がラスタ単位に行われているということです。

つまりラスタスクロールというのは，ラスタごとに画面表示開始位置を設定することにほかなりません。通常のスクロールは表示位置の変更を帰線期間内に行うのですが，これを任意のラスタを表示する直前に書き換え，また元に戻すとラスタ1本だけスクロールさせることができます。

ここで問題になるのが“ラスタ表示のタイミング”です。たとえば画面全体が512本のラスタで構成されているとして，10本目から20本目のラスタだけ表示開始位置をずらそうと思っても，いつ10本目のラスタを表示して，いつ20本目のラスタを表示し終わったか (21本目のラスタを表示し始めたか) がわからなければどうしようもありません。これを検出するのにラスタ割り込みを使います。詳細はプログラマーズマニュアルを参照してください。

ここではラスタスクロールのサンプルプログラムを紹介します。なにか絵を表示してからこのプログラムを実行してみてください。ESCで終了します。ゲームで似たようなものを見たことがあるでしょう。

リスト1

```
  1: *
  2: * ラスタスクロール サンプル
  3: *
  4:         .include        iocscall.mac
  5:         .include        doscall.mac
  6:
  7:         .text
  8:         .even
  9:
 10: wait = 12
 11:
 12:         clr.l   -(sp)
 13:         DOS     _SUPER          * スーパーバイザ
 14:         addq.l  #4,sp
 15:         move.l  d0,ssp_save     * SSP保存
 16:
 17:         move.w  #200-1,d2
 18: loop1:  move.w  #9000-1,d1      * ヘッドが上がるのを待つため
 19: loop2:  dbf     d1,loop2        * 約2秒空ループを回す
 20:         dbf     d2,loop1
 21:
 22:         move.w  #wait,count     * ウェイトカウンタ
 23:         move.w  #32-1,d1
 24:         lea.l   rasno,a1
 25: init_loop:
 26:         clr.w   (a1)+           * ワーク初期化
 27:         dbf     d1,init_loop
 28:
 29:         move.w  #$2600,sr       * 割り込み禁止
 30:                                 * 割り込みマスク状態を保存
 31:         move.l  $e88012,mfp_save
 32:         move.l  #$005e_0004,$e88012
 33:         clr.w   d1              * 割り込むラスタ番号
 34:         lea.l   rasint,a1       * 割り込みアドレス
 35:         IOCS    _CRTCRAS        * ラスタ割り込み設定
 36:         move.w  #$2000,sr       * 割り込み許可
 37: chkesc:
 38:         bsr     make_data       * ラスタスクロール用のデータ作成
 39:         clr.w   d1
 40:         IOCS    _BITSNS         * キーの押下状態を調べる
 41:         btst.l  #1,d0
 42:         beq     chkesc          * ESCが押されていない
 43:
 44:         move.w  #$2600,sr       * 割り込み禁止
 45:         clr.l   a1
 46:         IOCS    _CRTCRAS        * ラスタ割り込み解除
 47:         move.l  mfp_save,$e88012
 48:         move.w  #$2000,sr       * 割り込み許可
 49:
 50:                                 * スクロールレジスタクリア
 51:         clr.w   $e80018         * スクリーン0
 52:         clr.w   $e8001c         *           1
 53:         clr.w   $e80020         *           2
 54:         clr.w   $e80024         *           3
 55:
 56:         move.l  ssp_save,-(sp)
 57:         DOS     _SUPER          * ユーザー
 58:         addq.l  #4,sp
 59:
 60:         clr.l   -(sp)
 61:         DOS     _KFLUSH         * キーバッファをクリア
 62:         addq.l  #4,sp
 63:
 64:         DOS     _EXIT           * 終了
 65:
 66: make_data:
 67:         btst.b  #4,$e88001
 68:         bne     make_data_rts   * 垂直表示期間だ
 69:         subq.w  #1,count        * カウンタを1減らす
 70:         bne     make_data_rts   * カウンタが0じゃない
 71:         move.w  #wait,count     * ウェイトカウンタ設定
 72:         lea.l   scrool_dot(pc),a0
 73:         lea.l   rasno(pc),a1
 74:         move.w  #32-1,d0        * 512/16=32個分
 75: make_data2:
 76:         move.w  (a0)+,d1
 77:         add.w   d1,(a1)+        * スクロールデータ更新
 78:         dbf     d0,make_data2
 79: make_data_rts:
 80:         rts
 81:
 82: ********************************
 83: *       ラスタ割り込み処理
 84: ********************************
 85: rasint:
 86:         movem.l d0-d1/a0,-(sp)  * レジスタ保存
 87:         move.w  raster(pc),d0
 88:         move.w  d0,d1           * ワークにコピー
 89:         lsr.w   #4,d0           * ラスタ番号/16
 90:         add.w   d0,d0
 91:         lea.l   rasno(pc),a0
 92:         move.w  (a0,d0.w),d0    * d0 = x座標表示開始位置
 93:         move.w  d0,$e80018      * スクリーン0
 94:         move.w  d0,$e8001c      *           1
 95:         move.w  d0,$e80020      *           2
 96:         move.w  d0,$e80024      *           3
 97:         add.w   #16,d1          * 次に割り込むラスタ番号
 98:         andi.w  #511,d1         * ラスタ番号を0～511にする
 99: rasint2:
100:         move.w  d1,raster       * ラスタ番号更新
101:         add.w   #$28,d1         * ラスタ番号補整
102:         move.w  d1,$e80012      * 次に割り込むラスタ番号
103:         movem.l (sp)+,d0-d1/a0  * レジスタ復帰
104:         rte                     * 割り込み終了
105:
106:         .data
107:
108: scrool_dot:                     * ラスタ番号
109:         dc.w    1,1,1,1,1,1,1,1         * 0-7
110:         dc.w    1,1,1,1,1,1,1,1         * 8-15
111:         dc.w    1,1,1,1,1,1,1,1         * 16-23
112:         dc.w    1,2,3,4,5,6,7,8         * 24-31
113: rasno:
114:         ds.w    32              * 表示開始位置格納領域
115:
116: count:
117:         dc.w    0
118: raster:
119:         dc.w    0
120: mfp_save:
121:         ds.l    1
122: ssp_save:
123:         ds.l    1
124:
125:         .end
126:
```

なお，このプログラムはバグが取れるまでハードディスク上で実行しないほうがいいと思います（私はデバッグ中にハードディスクを飛ばしました）。もし，このプログラムを使ってなんらかの損害を被っても私は責任を持てません。個人の責任において実行してください。

ではプログラムの解説をしましょう。まず17～20行目で空ループを約2秒回していますが，これはこのプログラムを内蔵ディスクドライブから読み込んで実行する場合を考えてのことです。通常X68000の内蔵ドライブはフロッピーからデータを読み込んだあと，約2秒間モータが回転しています。その間アクセスランプは赤色に点灯しています。X68000ではその2秒間を測るのにタイマーC割り込みを使っているのですが，この割り込みを32行でマスクしています。ですからプログラム実行から2秒たたないうちに32行を実行してしまうと，アクセスランプが赤く点灯したままになってしまいます。これはちょっとやばい感じがするので，このようなループを設けて時間稼ぎをしているというわけです。

このようにラスタ割り込みに関係のない割り込みをマスクしているのは，割り込み中に効率よくCPUを使うようにするためです。このプログラムではFM音源の割り込みも禁止していますのでラスタスクロール中の演奏はできません。ちなみに32行を，

move.l #$005e_000c，～

とすれば，FM音源との同時演奏も可能ですが，画面がちらつきます。これは私の技術力の低さの表れです。不満があったら自分で改良してみてください。

プログラムのメインルーチンはラベルchkescから42行です。ここではESCキーが押されるまでループを回しています。ループ中で呼び出しているmake_dataは，ラスタスクロールに必要なデータをワークに設定するサブルーチンです。make_dataでは最初に垂直帰線期間かどうかを調べ，そうでなければただちに処理を中止します。垂直表示期間中は画面の書き換えが行われているので，その間にラスタスクロールに必要なデータを書き換えてしまうと画面がぶれてしまうからです。

さらに垂直帰線期間中であってもラベルcountの値を1減らして0でなければ，やはり処理を中止します。countの初期値は10行のシンボルwaitで設定しています。この値を大きくするとスクロール速度が遅くなり，小さくすると速くなります（理由は自分で考えてみてください）。

さてラベルrasintからが割り込み処理ルーチンです。ここではテーブルrasnoから表示開始位置を取り出し(92行)，93～96行で実際にCRTCに表示開始位置を設定します。97行で次のラスタ割り込みのラスタ番号を設定しています。ここでは16を加えているので，16ドットごとにラスタ割り込みがかかるようになっています（つまり，0,16,32,64……480,496本目のラスタを走査するごとに割り込みがかかる)。

質問にもありましたが，このように割り込むラスタ番号を次々と変えていく場合はIOCSコールを使わずに，CRTCに直接割り込むラスタ番号を設定します（102行)。設定する前に$28を足していますが，これは決まった形ですから気にしないでください。また，ラスタごとに割り込みアドレスを変更する場合は，割り込み処理ルーチン内で$138_H$番地に次の割り込みアドレスを設定します。$138_H$番地はラスタ割り込みのベクタアドレスです。たとえば87行に，

move.l #割り込みアドレス,$138.w

を挿入すれば，次にラスタ割り込みがかかったときには87行で指定した割り込みアドレスに処理が移されます。（影山　裕昭）

**TeXに関する特集記事を興味深く読ませていただきました。ところが最後に掲載された出力を見てビックリ。明朝，ゴシックは当然としても，毛筆書体，教科書体まで出力できるとは。いったいこれはどのようにして出力したのですか。　香川県　高島　芳宏**

書体倶楽部として発売されているフォントを使うには，次の4つのステップが必要です。

1）書体クラブを購入し，ハードディスクにインストール

2）TeXでフォントを管理しているfontmanに，このファイルの使用を指示

具体的には，texstart.batなどのバッチファイルでfontmanに与えているSYSファイル(これはテキストファイル)に，

font = 〔追加フォントファイル〕

という行をED.Xなどで追加します。

3）追加フォント用.tfmファイルの作成

毛筆10ポイント用の.tfmファイルを作成するなら，日本語の.tfmファイルが格納されたディレクトリに移り，

A>copy dnpmn10.tfm mou10.tfm

とします。

以下の作業を続けるには，fontmanを再起動する必要があります。再起動したら，

A>fontman -i

として，追加フォントに与えられたIDを確認してください。ここでは仮に$0AというIDが与えられたとします。

4）追加フォント用.pkファイルの作成

適当なディレクトリを作成し，ここで，

A>genpk -z -10only $0A 118 dnpmn

として解像度118ドットのdnpmn10.pkを作成します。できたらそれをmou10.pkにリネームし，解像度118ドットの日本語PKファイルが収められているディレクトリにコピーしてください。以後，

¥font¥mou=mou10

のようにして使用することができます。解像度118ドットのフォントは画面表示用です。印刷するにはプリンタの解像度(180あるいは360)に合わせてPKファイルを作成し，それぞれのディレクトリに格納してください。また，magstepを使用する場合は129，142，170など，それぞれのmagstepに合わせた解像度のフォントを作成し，該当するディレクトリに格納する必要があります。（泉　大介）

**質問にお答えします**

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を上げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに回答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので，電話番号も明記してくださいね。

**宛先：〒108 東京都港区高輪2-19-13**
**NS高輪ビル**
**ソフトバンク株式会社出版部**
**「Oh! X質問箱」係**

# FROM READERS TO THE EDITOR

そろそろ動きやすい季節になりました。文化祭,体育祭など楽しい行事もたくさんあることでしょうから,体力の続くかぎりがんばりましょう。何か新しいことを始めるもよし,秋の夜長にひとり思い出に浸るのもいいかもね。

◆8月号の特集はよかったですね。ちょうど私自身,印刷について興味があっただけに非常に参考になることが多かったです。なかでも「High Fidelityへの挑戦」にはびっくり。いやはや,同じようなことを考えている人っているもんなんですね。　井上　博嗣(22)三重県

◆ここしばらく,CZ-8PC4がカラープリンタだということを忘れていたように思います。そんなところに8月号の特集。中野さんの熱意には頭が下がりますね。それにしてもCZ-8PC4の48ドットモードは色のりが悪い。潜在能力はかなり高いと思うのだが……。現在,しきい値を変更したりして試しています（ああっ,カラーリボンが)。　佐藤　充浩(20)長崎県

プリンタを持っていない人には,ちょっと酷な特集だったかな。

◆48ドットプリンタもCZ-8PC5で2代目だというのに,いまだ対応する市販ソフトがないのはどういうことなのだろうか。ところで,8月号の浜崎さんの記事はなかなかよかったけど,CZ-8PC4/5がついでという感じでちょっと残念。次回のプリンタ特集では,CZ-8PC4/5専用のハードコピープログラムを期待したいですね。　福田　秀昭(22)千葉県

せっかくの周辺機器ですから,福田さんもがんばって120%の性能を引き出すような使い方を目指しましょう。

◆8月号の特集はとてもきれいで感動しました。ところで,編集部には紙の無駄が気になる方がいたようですが,僕もプリントアウトやコピーで紙の無駄を大量に作ってしまい,後悔することがよくあります。そんなときには,紙供養にかぎります。やり方は簡単。無駄になった紙の前で「え,ごめんよう。それ,ごめんよう。もひとつ,ごめんよう……」と紙様にあやまりましょう。「たたりがありませんように」とね。　藤戸　正道(23)東京都

なんかいいかげんだなあ。

◆ 8月号の表紙のミノタウロスちゃんて“いろっぽい”と思いませんか？　特に腰のあたりがね。　加藤　恵吾(15)愛知県

惚れちゃった？

◆A列車で行こうIIIは面白い。ある程度,街を発展させてビルが立ち始めたときからがとてもいい。夕方,夜へと時が移り,仕事帰りであろう人々を乗せたAR-IIIを見ていると,なんだか電車の窓から外を眺めている気になる。そして,深夜になっても消えないビルの灯りは日本を感じさせてくれる。5時から男のための店,繁華街のネオンサインもあったらもっとよかったかな。また,中途半端な開発が両親の田舎を思い出させてくれるのもいいね。　田宮　浩(17)千葉県

AM5：00から男の店もあったらもっと現実的でいいね。

◆ハードウェア工作が本当に走り出したので,結構うれしいです。こうなったら,いっそのこと「しっぽつき」はやめにしましょう。秋月のZ80ボードでもってスタンドアローンにして,そのついでにROMライタの製作まで一気にもっていってほしいですね。

ハードウェアから自作して,8Kバイトぐらいのプログラムを書いて喜んでいる私としては,X68000の2Mや8Mバイトなどという話はとてもついていけない。ということで1年に1度は,ハードバリバリの特集があるといいな。　細谷　俊彦(19)神奈川県

新しく走りだしたハードウェア工作入門ですが,お気に召しましたか？

◆思い切って生中継68を買いました。画面は期待どおりの臨場感があり,すごいですね。ただ,操作性が一般的な野球ゲームと一部違っているので,「これは,むむむっめめめっ」となりましたが,慣れてくるとやっぱりいいソフトだな,と思い直しました。しかし,いまだに慌てると何塁に投げるかとっさにわからないのでそれだけはなんとかならないかな。きっと生中継68IIを出してくれるだろうから,そのときには……。ちなみに,まだ2人プレイとか新チーム結成はやってません。　宮武　克昌(24)大阪府

操作を選択できたらよかったですね。

◆X68000版イースの発売を知ってから慌てて“夜明け・イースバージョン……いえ,イースアレンジ”を打ち込んでみました。う～ん,西川さんってやっぱりすごい！　おかげでメロディが頭から離れません。でも,赤毛のアドルが貧乏旗本の三男坊だったら嫌だよう。じゃあ,西川さんの次回作に期待しています。　岩瀬　貴代美(19)福岡県

天下のヒーローが貧乏旗本の三男坊ではかっこがつきませんね。

◆まず,目次を見て「おおぉ,進藤さんだあ」。そして3時間後に打ち込み終わってから「なんかいまいち」って思ってしまいました。私もぜいたくになったものだ。自分でプログラムする分にはあそこまでできないのに,人にはなんくせつけたがるとは悪い奴だなあ。あたしってば。そうそう,とーとーイラストが載ってしまいましたなあ……。う～ん,やっぱりペンが必要かあ（まだ持ってない)。今度大学生協で買ってこよう。　谷口　有香(22)北海道

力作を期待してますよ（イラスト&プログラム)。

◆「7月号149ページのハミダシは……」の田中です。いやあ,「SIDE STREET」よかったですよ。3時間ほどで打ち込み終わり,さっそく聞いてみるとやっぱり進藤君でしたねえ。　田中　明仁(19)大阪府

◀丸藤　俊之　神奈川県
電気釜に入ってるなんて,おもしろい女の子ですね。と冗談はこれぐらいにして,ミニATみたいなボナンザがかわいいなあ。

◀小川　裕美　山口県
少し絵柄が丸くなりましたね。それにしてもこんなかわいいFIGHTERに出会ったらへろへろになっちゃいそう。でも,性格はきつかったりして。

やっぱり進藤君なんですよ。

◆今日は19日だからOh!Xを買いにいこうと思い，本屋へ行ったところがない！　考えること数分，お下劣本と間違えられたか，と思いそちらを見てもない。そして，あきらめかけた頃，なんと音楽雑誌のところにFM-STATIONと一緒に置いてあるではないか。でも，なぜ？　それはOh!Xの隣にあったOh!FMによって明らかになった。う〜ん，納得。　黒畑　喜弘(18)富山県

いくらミノタウロスがいろっぱいっからってお下劣本と一緒になるとは思えないけど。いまだにこういうことってあるんですね。

◆ある日，電車の中でOh!Xのページをめくりつつ語らう青年たちを見かけた。Oh!Xもメジャーになったものだなあ，と感慨にふけっているとその感慨をかき消すように，あることわざが僕の頭をよぎった。類は友を呼ぶ……。

河野　太郎(18)東京都

そしてできあがる最強の集団ってか？

◆近所の本屋で8月号に付録として，「RURUBU8月号特別付録」と書かれた北海道の地図が付いていました。いったい何を考えてOh!Xにこれを入れたのかまったく理解できません。

加藤　聡(22)東京都

いいなあ，北海道って1度行ってみたかったんですよ（そういう問題じゃないって)。

◆なあにい！　「由貴ちゃんは俺のもの」だとお。まあいい，許そう。そのかわり「レビアさんは私のもの」だ！　赤木　豊和(23)兵庫県

そんなことは，(で）氏が許してもこの私が許しません。レビアは俺のもんだ。

◆DEFEAT2をやりましたよ。最初はノーコンティニューでクリアしようとしたけどかなり難しくて，結局あきらめてしまいました。地上の敵を破壊しようとすると敵戦闘機に狙われたりして，出現タイミングがぴったしなので非常にやっかいだったな。そうこうしているうちに“C”キーを押しまくってなんとかエンディングを迎えることができました。最後のボスは手強かったけど，とにかく燃えました。

三沢　弘之(20)神奈川県

追加面希望のハガキもちらほら，浅野君見てる？

◆X68000を買って間もない頃，シャープの受付のおねーさまに「バスエラーってなんですか？」と20分間聞いていたことがありました。そんな僕でもいまではアセンブラを使うまで成長しました。すべて村田氏の連載のおかげ。本当にありがとうございます。これからもず〜っと続けてね。　堂領　輝昌(17)宮城県

君も大人になったんだね。

◆いまの世の中，ちょいと新しいメディアに対する欲求が弱いですね。どんどん出てきたものが満腹になっている感じがするんです。まあ，こういったものには波があるものだから，そのうち変わっていくでしょうけど，シャープさんは新たな欲求の波に耐えるものを用意していますか？　高橋　政秀(18)東京都

何が出てくるか本当に楽しみですね。

◀中川　和之　埼玉県
予定は未定，確定にあらず。ということでがんばってX68000を買おう。10代最後の夏，有意義に過ごすことができましたか？

◀大島　貴成　東京都
僕もいろいろなフォントを使ってみましたが，結局ROMフォントに落ち着いてしまいました。下手に飾りすぎると見づらくなっちゃうんですよね。

◆現在，文書処理は曲がり角にきています。会社の中では，一太郎派，Mac派，TeX派，自社文書派が乱立しているといったところでしょうか。最近はMac派が伸びてきています。日本人は表が好きなのでTeXはちょっとつらいようです。また，他人と文書をやりとりすることが多いため一太郎派は強力です（どこの会社にもある)。私はVz＋ATOK7に慣れてしまったので，MacやWnnには戻れません。今度，UPZシリーズに移行しようかと思うのですがどうでしょうか。こう考えるとFIXER＋SX-WINDOWは結構強いかもしれませんね。　臼渕　啓明(25)神奈川県

編集部の主流はWP.Xですが，最近（T）氏がSX-WINDOWに浮気しているみたい。

◆7月22日の朝に起きて，ふと気づいたのだがSX-WINDOWというのはX-WINDOWを右へローテートしたものだったのか。気をつけろ！　シャープはあなどれないぞ。轟原　正義(18)大阪府

このハガキを見たSX-WINDOWの開発スタッフが，どきっとするかそれとも高笑いをしているか，真実はどうなんだろう。

◆ブラインドタッチをマスターしようとしてキーボードを見ずに，ブラウン管を見つめすぎて視力が下がってしまった(トホホ)。皆さんも気をつけましょう。ところで，8月号の表紙はもしかして（で）さんですか？

佐渡　詩郎(15)石川県

ありゃりゃ，無理はしないでね。

◆ここ数年，夏になると海に潜りっぱなしでした（佐渡だもんね)。今年の夏は「うんと楽よ，家族を連れておいで」との甘い誘いについのせられて尾瀬に行ってきました。ところがいざ行ってみると「どこが楽なんだ！」。2歳の子供をかついで泥まみれになって歩いた林道。日頃運動不足の私にとって，よくもまあ生きて帰ってこれたものよ……。

友人の弁解いわく「楽しかった尾瀬ヶ原のことしか頭になくて，そこまでの道程はすっかり……」だそうです。確かに尾瀬沼も尾瀬ヶ原も素晴らしかった。だが，つい先日手に入れたX68000のマウスをゴリゴリしながらの四畳半のほうがいいよ〜っと林道では叫びたかった。

石塚　孝之(35)新潟県

これは友人にしてやられたって感じ。でも，たまの家族サービスですから少々の苦労は……ね。

◆何を隠そう尺八演奏を趣味としている私は，あまりコンピュータミュージック，特にMIDI関係には興味がありませんでした。しかし，9月に家内のバースデーを迎えることになった我が家で，プレゼントのネタになんとMIDIキーボードに白羽の矢がたったのです。ということで現在，MIDIキーボードを物色中。尺八なんかと演奏すれば名（迷？）演奏ができるかもしれないと密かに思っています。　河添　勲(30)千葉県

これを機会にコンピュータミュージックの世界へ冒険してみてはいかがですか？

◆こんにちは！　我が家の古代米（縄文時代からの赤米，黒米）の栽培は，その後，プランター式の田植えも終わり順調に成育中です（やや赤米が成育不良ですけど)。地面より長さ30cmくらいの苗が数十本，ベランダの光の中で輝いています。さて，秋にはどれだけの古代米が再生するか楽しみです。　迫田　賢一(40)大阪府

それだけ愛着を持って育てていると，収穫して食べてしまうのがかわいそうになったりして。

◆最近やっと18歳になりました。そして，友人たちは「おめでとう」の言葉の次に「借りてきてほしいビデオがあるんだけど」だって。見事に会員証まで作らされてしまいました。

奥山　浩司(18)三重県

ちなみに誰のビデオを借りたのかな？

◆7月から自動車教習所に通っているのだが教習所にはいろんな人がいる。信号無視をする人，コースを出て草むらを走る人など。以前，サイドブレーキを引きちぎった人やウインカーのレバーをへし折った人もいたそうだ。

福地　健(20)京都府

その他，半クラがわからずクラクションを半分の力で鳴らしたり，エンジンブレーキと言われて補助ブレーキを踏もうとした女性がいたようです。

◆この間，昼間からライトをつけっぱなしにしている，東京電力の車を見たがこういうことでいいのだろうか？　大橋　秀明(21)神奈川県

いくない。じゃない，よくないですね。

◆先日，ポテトチップスのCMを見て思ったんです。CMでは袋のマークを切りとって送ると，送り主に代わってお金を寄付するということですが，どうせなら切りとって送るのではなく袋ごと送るようにすれば，もっとエコロジーなのではないでしょうか。切りとったあとのものは捨ててしまうんだし，なんかよくわかりませんね。コカコーラの“7点集めて送ろう”も環境問題と関係ありそうな気がするんですが。

山崎　乾太郎(18)愛知県

もっともな感じもしますね。

◆ある暑い夜の出来事。机の前に座って足にできたかさぶたをいじっていた。しばらくすると血の固まりが取れた。意外に気持ちがいい。ふと見ると，机の上を小さなゴキブリがうろついていたので，そいつの鼻面に先ほどの血の固まりを投げてやると食らいついた。ゴキブリに餌を与えるなんて珍しくて面白いなあ。もしも，このゴキブリが人間の血の味を覚えてしまったら……。そのゴキブリはすぐにつぶした。ある暑い夏のことでした。　大久保　益幸(19)滋賀県

まあ，そんなことにはならないでしょうが一瞬，全身びっちり覆い隠すように群がり人肉を貪るゴキブリを想像して，気持ち悪くなってしまった。

◆暑い！　これからこの暑さがしばらく続くのかと思うと汗が止まらない。ただでさえ軽い体重がさらに軽くなってしまう(現在47kg)。これじゃあ体がもたん。なんとかしてくれい。

山口　大賀(18)愛媛県

でも，夏痩せする人ってちょびっとばかしうらやましいな。最近，おなかの肉が……。

◆大学に進学するため世田谷に下宿を始めてから3年目。下宿生活を始めていちばん困ったことが水のくささで，歯を磨くときも気になっていました。なにしろ，実家は岐阜の田舎なのでよけい東京の水のまずさが感じられるんです。ところが，3年になると同時にキャンパスが目黒区から文京区へ移って，自分の認識の甘さを知りました。世田谷の水はこれに比べれば……。ところで大阪の水はもっとまずいって本当なんですか？　箕浦　真(20)東京都

関西出身の3人に聞いてみました。答えは3人とも「琵琶湖の水はちょ～まずい」だそうです。

◆つ，ついにX68000XVIを買うことになりました。いきなり買うとは思いもよらなかったのですが，以前買っていた宝くじが100万円当たってるとは……。多少高くてもすぐに入手できる店を探し，CM-64で音楽をZ's STAFFでイラストを……X68000って楽しいな，と思う今日この頃でした。

秋定　貴文(17)兵庫県

う，うらやましすぎる！

◆こんにちは，空手初段の広田です。以前，なぜか市民大会の個人組手で優勝しまして，今度はインターハイの団体組手で全国大会へ行くことになりました(でも静岡だから暑そう)。Oh!Xはこんな御利益もあるのですね。私を知らない人は1991年3月号168ページを読んでみましょう。

広田　政則(17)北海道

僕もハガキ破りを鍛えて全国大会を目指すぞ。

◆コミックモーニングのゴールデンラッキーにはまってしまいました。1部に熱狂的なファンをもつ例の4コマ漫画です。まだ読んでいない人は1度読んでみることをお勧めします。単行本は2巻まで発売中ですがなかなか手に入らず，いまだに1巻が手に入らない。買って，読んで，ソリが合いさえすれば，あなたにバラ色の人生が訪れることを保証します。保証はしますが苦情は受け付けません。にやり。岡本　直樹(18)京都府

編集室でもライターの（浦）氏がはまっていたなあ。でも，現在最も人気が高い漫画は「寄生獣」みたい。

◆「おもひでぽろぽろ」を観ました。単なるレトロものとは違い，人はどう生きるべきか，というようなことを考えさせられる映画でした。また，現在の日本の農業が抱えている問題にも触れていて，農学部の自分にとっていい勉強になりました。でも，次回作は「ラピュタ」のような冒険活劇が観たいです。松本　正弘(21)京都府

これからもいい作品を作り続けてもらいたいですよね。

◆最近，思いがけずX68000ユーザーとお友達になることができた。で，さっそく遊びにいって「へっへっへ，さーて何をプログラムしようかな」と思ったはいいが，X68000の使い方なんていままでろくに勉強しなかったので何もできないことに気づいた。ここでゲームでもできればまだましなのだが，レビューさえ読み飛ばしていたのがたたり，もはやタイトルを見ても内容がわからない。情けなくもその日はゲーム音楽を聞かせてもらうにとどまった。今度はちゃんと予習するぞ。　金原　真也(23)宮城県

持ち主よりも使いこなせるようにがんばりましょう。でも，そんなことになったらせっかくの友達に嫌われちゃうかな。

◆僕の家は愛媛でも結構山のほうにあり，夏はあまり暑くなりません。だからクーラーなんぞもちろんなく，扇風機はあってもほとんど使う必要がなく，使うのはうちわだけです。ちなみに蚊帳があります。そして毎年ハエ取り紙を使っているところはそうないでしょう。だいたい最近ではハエ取り紙を知らない人もいるんじゃないでしょうか。　近藤　英二(20)愛媛県

蚊帳といえば僕が小学生の頃，毎年田舎に帰ると蚊帳を使い漁師さんごっこして，両親に怒られたなあ。

◆Oh!Xで市民権を得る資格は「16歳以上23歳未満」のようですね。STUDIO Xのコーナーに載っている人の大部分はこの範囲内ですから。ついでにいえばハミダシは「16歳以上24歳以下」のようです。よし，あと1年だ。

佐々木　亮(15)大分県

こらこら，勝手に年齢制限など設けないように。コンピュータが好きなら年齢は関係ありませんよ。

◆7,8,9月とゲーム機，パソコンともに買いたいゲームソフトが多く出てくる。しかし，その時期は仕事が忙しくて，とてもゲームなどやってられなくなってしまう。1日1時間は必ずゲームをやっていた私としては，ゲームができないと非常に悲しい。そんなときはいつもパッケージを見ながら心の中で，必ずクリアしてみせるからな，とつぶやきながら寝ることにしている。

今井　明広(22)愛知県

えらい！

◆最近，電車やバスに乗っているとき，ふと窓から外を見て「おお～，なんてリアルな多重スクロールなんだ」と思ってしまう僕はシューティングゲームのやりすぎでしょうか。

藤田　伸夫(24)新潟県

これはかなり重症です。はやく一般人として更生することを祈ってます（人のことはいえないけど）。

◆夏のくそ暑い中，皆様いかがお過ごしでしょうか。私はといえば，下宿先で家ダニが大量発生してむちゃくちゃな思いをしました。6月末の梅雨の間にあった，うそのように晴れた暑い日に，右手のひじから手首の間を集中的に60～70カ所を刺されたのです。んで，それもおさまりかけたおととい，またもや久々のピーカンの日に畳の上で寝ていたら，今度は左手を20数カ所刺されてしまいました。おまけに脇腹まで刺されて非

▲橋本　浩志　岡山県
ZOOMといえばネコが有名ですが，このイラストではなぜかナセルがかっこいいぞ。でも最新作のファランクスの影が薄いなあ。

▲山岸　稔　埼玉県
あ，あぶない。正視していると電波が飛んできそうなイラストだ。でも目を離そうとしてもついつい目が……。僕はなにもしていないぞ！

常にヒサンです。皆さんも部屋はこまめに掃除しましょうね。っていってももう遅いかな。

松本　康弘(24)広島県

想像しただけでかゆくなりそう。うえ，鳥肌立ってきちゃった。

◆「Miss GEOS」は，昔好きだった人の後ろ姿に似ていて胸を焼かれた。

佐藤　健太郎(21)東京都

う～ん，青春だねえ。

◆先日我が校でFM TOWNSを30台ほど購入した。自由に使えることを楽しみにしているのだが，なかなか使わせてくれない。コンピュータクラブを設立してくれそうにもないし，このままではいまの1年生が3年生になって，情報処理の授業を始めるまで電源すら入れないような気がしてきた。いったい，何を考えているんだか。

林　裕司(15)福島県

それは，もったいない話ですね。あるものは活用しなくちゃ。

◆あるコンピュータメーカーを見学しにいったとき，アンケートを書かされた。どんなコンピュータを使ったことがありますか？　という質問に，思わず大きくMZ-700と最初に書いてしまった。

荻原　豊隆(22)長野県

清く正しいシャープユーザーの鏡ですね。えらいえらい。

◆7月20日，数日前から友人宅でやっていたブラックオニキスが解けた。その前はアスピックです。経験値稼ぎが面倒臭かったけどなんとかエンディングまで到達。なかなか意外な結末で面白かった。もしも，途中であきらめている人がいたら，がんばって最後まで到達してみてください。さあ，次はファンタジアンだ(もちろん全部テープ版)。

本田　和正(17)愛媛県

秋の夜長は懐かしいゲームを引っ張り出して遊ぶにかぎりますね。

◆スキー部の合宿で立山に8泊9日の予定で行ってきました。40kgのザックを背負いながら板を持っての電車＋歩きで行きましたが，1日だけ滑って次の日は大雨＋強風でテントが破れ，雨でさらに重くなったザックを背負いながらしょうがなく下山。もう，泣きそうになった合宿でした。

米田　公昭(19)東京都

ごくろうさまです。来年はもう少し楽しめるといいですね。

◆7月号の知能機械概論を読んで，さっそく休講の時間にNeXTをいじりにいった。初期型だったためか，結構遅く感じられました。そして，いく

▲石田　伯仁　神奈川県

PC-8801というとその昔，レリクスを友達と一緒に遊んだことがあるなあ。でも僕はシンプルでなにもなかったMZ-2000がいちばん好きです。

らアイコンがあるとはいえ英文を読めないと理解できないことがわかった。ピンボールはマウス一発だが，フライトシミュレータはキー配置がちんぷんかんぷんだったからなあ（なんのためにNeXTを使っとるんじゃ）。

寺門　修司(20)兵庫県

そんなもったいない使い方をしてると，スティーブン・ジョブズが泣きますよ。

# ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できない場合もあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

## 仲間

★当サークル「JSCX68」ではX68000ユーザーの会員を募集しています。活動内容は月1回のディスク会報による情報交換やPDSの配布などです。特にPDSは通信をされていない方のために充実しています。興味のある方は62円切手を同封のうえ，住所，氏名を明記して下記の住所までお送りください。サンプルディスクをご希望の方は500円分の郵便為替，もしくは切手を同封してください。〒770　徳島県徳島市助任本町1-10-1 ウエーブ21　202号室　石本　良信(18)

## 売ります

★X68000用2Mバイト増設RAMボード「CZ-6BE2」を20,000円で売ります。箱，説明書あり。完動品です。3カ月間使用しました。連絡は往復ハガキでお願いします。〒230　神奈川県横浜市鶴見区鶴見中央3-20-9-1103　金田　宏卓(18)

★2HDディスクドライブ「FD55-GF」を4,000円で売ります。X1turboシリーズに接続可。動作確認済み。2台ありますので希望の台数と使用しているコンピュータの機種名を明記のうえ往復ハガキで連絡してください。動くまでできるかぎりサポートします。〒225　神奈川県横浜市緑区元石川町5264　中井　厚(21)

★JX-450（カラーイメージスキャナ）を送料込み180,000円で売ります。これは中古で買ったものです。連絡は往復ハガキで。〒260　千葉県千葉市穴川4-5-20　森　康夫(22)

★X1用ディスクドライブ「CZ-503F」を20,000円ぐらいで売ります。箱，付属品，説明書あり。完動品。連絡は往復ハガキで。〒630-22　奈良県山野辺郡山添村勝原579　井場　秀樹(19)

★「CZ-8NS1」(カラーイメージスキャナ)を70,000円以上で。高く買ってくれる人優先します。完動美品，説明書，ケーブルつき。詳細および連絡は往復ハガキで。〒230　神奈川県横浜市鶴見区市場大和町2-23-302　山口　理(19)

★FM音源ボード「CZ-8BS1」を送料込み7,000円で売ります。箱，付属品，マニュアルあり。取りに来られる方には送料分を引きます。連絡は往復ハガキで。〒245　神奈川県横浜市戸塚区原宿町539-7　浅井　和彦(47)

## 買います

★BNNのPC-8801VAテクニカルマニュアルを送料込み10,000円で買います。多少の汚れは可，切り抜きは不可。連絡は官製ハガキでお願いします。〒700　岡山県岡山市東古松213　板野正男方　藤塚　弘樹(25)

### バックナンバー

★Oh!X1989年9，10月号，1990年1，3，7月号を各1,200円（送料込み）で買います。切り抜きやひどい汚れがないものにかぎります（付録なしも不可）。連絡は官製ハガキで。〒355-02　埼玉県比企郡嵐山町平沢35-10　加賀谷　匠(16)

★Oh!X1990年6，9，10月号を各1,200円程度（送料込み）で買います。切り抜きあり，付録ディスクなしは不可。傷，多少の汚れはかまいません。連絡は官製ハガキで。〒950　新潟県新潟市鳥屋町1-3-10　小林　貴秀(15)

★Oh!X1990年6月号を2,000円（送料込み）で買います。付録ディスクなしでもけっこうです。切り抜き等は不可。連絡は官製ハガキで。〒285　千葉県佐倉市王子台5-8-13　芳賀　浩史(22)

★Oh!Xの1990年6，7月号を定価で譲ってください。切り抜きは不可。送料はこちらでもちます。連絡は官製ハガキで。〒563　大阪府池田市畑3-1-24　玉井　晃伯(20)

★Oh!X1988年12月号を送料込み1,200円で買います。MusicBASICと特集が読めれば少々の汚れは可。連絡は官製ハガキで。〒761-04　香川県高松市三谷町884-4　西池　陽一(13)

▶9月号の"ぼくらの掲示板"でバックナンバーを求めていた栗原です。北海道登別の大塚さん，住所がどこにも書いておらず困っています。これを読んでいたら大至急連絡してください。

栗原　景(20)東京都

# 編集室から from E・D・I・T・O・R

## DRIVE ON

このコーナーでは，本誌年間モニタの方々の意見を紹介しています。今月は8月号の内容に関するレポートです。

●昔々，私がXIturboをメインで使っていた頃，一時期「IO-700」が無性にほしかったことがあります。もともとデジタル8色のグラフィックでしたから，カラー8色のインクジェットプリンタは画面上に表現されたものを（一応）忠実に紙の上に表現できました。それに比べて，現在ではブラウン管における表現力は飛躍的に発達しましたが，プリンタのほうが追いついていないわけです。それを補うのが8月号の特集でした。少なくとも現行のプリンタではほぼ最高の表現力でしょう。私はといえば，安さにつられて買った「CZ-8PC4」があるのみ。個人ユースでは十分なプリンタですが，いまさらながらに「IO-735X」がほしくなってしまいました。金さえあればなあ。

中村　健(21) X68000 ACE,XIturbo,MSX2＋　埼玉県

●新製品紹介の「サウンドキャンバス」，なかなかよさそうな音源ですね。本体の写真を見るかぎり，ローランドもなかなか気を利かした設計をしている（前面のヘッドホン端子やMIDI INなど）ようなので，使いやすいのではないでしょうか。MTやCMシリーズとの互換性において少々疑問があるものの，Uシリーズと同じRS-PCMを使っているようだ，とのことで「音」には安心できそうですね。

まあ，とにかく現状ではまだまだゲームのMIDI対応のBGMにしても発展途上だと思う（MTやCMのLA音源でもまだまだ内蔵音色が主体ですから）ので，ぜひ頑張っていってほしいと思いますし，環境もどんどん整っていってくれればいいと思います。音色の内容については記事ではあまり詳しく述べられていませんでしたが，MTとCMシリーズを凌ぐ音色数を持っているだけに早くその音色を聞きたいものです。

それと九十九電機の3.5インチのFDDですが，最近はMu-1のニューバージョン等でMIDIの標準ファイルがサポートされているようですから，X68000でMIDIファイルを3.5インチの2DDへ出力してちょうど発売になったシーケンサ，サウンドブラッシュに読ませるというのもいいかもしれませんね。まあ，少々不便なところもあるようですけど，「X68000」用として発売してくれたのもなにかの縁，九十九さんのサポートに期待しつつ，どんどん活用する方法を探すのも面白いと思います。

前田　秀樹(17) X68000 PRO,MSX/2　京都府

●「X68000マシン語プログラミング」は今回は拡大縮小ということで，それほど複雑なことはしていないようですが，次回の回転となるとどうなるのでしょうか（この意見が載る号ではもう終わっているでしょうが……）。回転と聞くと，私の場合，回転行列を使ったアルゴリズムぐらいしかうかびませんが，はたしてどのようなアルゴリズムを紹介してくれるのか，いまからとても楽しみです。将来はMAGICと融合させて……，などと考えているのは私だけでしょうか。

片木　章人(18) X68000　大阪府

●私は知らない曲を聞くのが好きなので，「Oh!X LIVE in'91」は毎月楽しみにしています。実際9割は知らない曲なので，打ち込みながらワクワクしています。LIVEの曲はほとんどがOPMD対応で，ドラムの音も元気いいのがほとんどですが，贅沢な私はやっぱりおとなしい曲も聞きたいと思います。ゲームミュージックや歌謡曲ばかりでは大人と呼ばれる方々は満足しないかもしれませんから，ここらへんでクラシックやジャズなんかの特集を組むのもいいかと思います。でも，当の私が以前LIVEに送ったのもゲームミュージックでしたね（笑）。まあ，今度がんばって「TAKE THE "A" TRAIN」でも作りましょうかね。でも，やはりFM音源はジャズとかには合わないのかしら。

谷口　有香(22) X68000　北海道

●8月号の特集「印刷の世界へ」は，以前のMIDI特集と同じく，機器のない歯がゆさを感じました。でも，それゆえにこれからどのようなタイプのものを選ぶべきかの指針になることでしょう。それにしても見開きページには，「マイッタナア，モウ」です。まぎれもなくプリンタでの出力であるにもかかわらず，あの福原さんの描き出す質感がここまで！　うーん，いい味出ている。女の子も安産型みたいだし。安心安心。何が？

弦元　達也(20) X68000 ACE-HD　香川県

●「AFTER REVIEW」は本当にユーザーの率直な意見を聞くことができるので参考になる。「THE SOFTOUCH」のほうでも，かなり辛い批評があるのである程度はわかるが，やっぱりメインは紹介であるので，もうひとつ，つっこめない気もする。それに比べて，このコーナーではすべて本音，しかも遠慮なくズバズバというので，身に染みてその雰囲気が伝わってくる。どうしても写真とかだけではわからないこともあるし，これは買うときに大変参考になる。

国政　寛(20) X68000, PC-6001　大阪府

●私はつい最近になって，「A>」を「エープロンプト」と読めるようになりました。「A>」も「エープロンプト」も別々に知っていました。でも，同じだとは思わなくて，「A>」を「エーサンカク」と読んでいたのです。

こんな当たり前のことすらわからないものもいるんです。特にパソコンって慣用句が多いから，とっつきにくかったりするんですよね。

野原　志貴乃(29) X68000 ACE-HD,XIturbo　埼玉県

## ごめんなさいのコーナー

**9月号 ぺんぎん情報コーナー**

P.165 北海道コンピュータグラフィックス協会の連絡先が古い電話番号になっていました。正しい電話番号は☎011(210)9221です。たいへんご迷惑をかけました。

**9月号 Oh!X LIVE in '91**

P.88 リストの一部に見にくい個所がありました。以下に再掲載します。

```
4030　c(15)="v15a+16&b-8.b-b-4b-4a&
ab-4<c4.>……
```

**9月号 Oh!X THE USER'S WORKS**

P.28 O/S softwareの住所が不正確でした。正しくは「埼玉県久喜市南2-2-7」です。ご迷惑をおかけしました。

**バグに関するお問い合わせは**
**☎03(5488)1311(直通)**
**月～金曜日16：00～18：00**

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます。入力法，操作法などはマニュアルをよくお読みください。

また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません。ご了承ください。

# ハガキはみんな生きている

▼皆さんは現在，どのような言語をお使いでしょうか。今年の6月号のアンケート分析では，BASICが過半数で優位，C言語とアセンブラがほぼ同じレベルでやや少数，という結果が出ています。C言語がポピュラーな言語になってきた。BASICもコンパイルすればそこそこのスピードになる。とはいえ，まだまだアセンブラに対する関心度が低くなってきたとは思えません。アセンブラ，すなわちマシン語はすべての言語の根底に流れるものとして，いつまでも人々の征服欲の対象でありつづけることでしょう。今回の特集「マシン語との邂逅」は，他言語からアセンブラへと移行しようという方々を主な対象として考えています。皆さんもコンピュータという世界のもっと奥深くを覗いてみませんか。

▼読者の皆さんからのアンケートハガキを読んでいると，実にさまざまなことを知ることができます。最近あったこと，巷で流れている噂，記事内容の善し悪し，そしてもっとも重要な，Oh!Xに対しての率直な意見。編集者はもちろんのこと，ライターの方々も暇を見つけては，ハガキを隅から隅まで貪るように読んでいます。「なるほど！」という意見もあれば，不可解な意見もあります。しかし，すべてなんらかのかたちで記事に反映されています。アンケートハガキは読者とスタッフを結ぶ，重要なパイプなのです。ですから，これからもなんでも思ったことを記入して，こちらに送り返してください。まあ，あまりにも変なのはちょっと困りますけどね。

▼ではお約束どおりに，8月号のアンケートハガキのプレゼントNo.のところに0番と記入された方々のうち，つぎの10名の皆さんにTシャツを差し上げます。

矢野啓介（北海道），西山大輔，山崎幹生（新潟県），柴田千春（宮城県），広瀬良一（茨城県），宮川秀昭（千葉県），加藤正栄（神奈川県），福岡尚久(愛知県)，田中拓也，勇崎昌宏（京都府）

おめでとうございました。

**投稿応募要領**

●原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。

●プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。

●ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。

●投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

**あて先**

〒108 東京都港区高輪2-19-13 NS高輪ビル

ソフトバンク出版部

Oh!X「㊢㊀㊣㊔」係

# SHIFT・BREAK

▶2週間ほど帰省してきました。暇潰しに昔のOh!MZ（おおっ）をぱらぱらと読み返していると，まだ読者してる頃の自分の名前がちらほら……。こんな時代もあったっちゃねー（博多弁）と懐古すると同時に，時が流れてもやってることがほとんど同じ自分を発見して，こうなりゃ一生同じことをやり通そうと決意を新たにした私でした。　原稿描いて22年の（哲）

▶学生なので夏には夏休みというものがある。毎日数時間を学校で過ごす私にとって天国のような日々が待っていたはずなのだが，いざ始まってみると蟻の涙ほども楽になっていやしない。よくよく考えてみると，どうやら寝すぎに問題があるらしいのだが，早起きの方法を知らない私は，今日も昼過ぎにゆっくりと起きるのであった。（八）

▶先月買ったメガドライブのマーベルランドをクリアしました！　最近はアクションゲームにもエンディングがあるんですね。あのパロディウスも“北の国から”面でつまずいて，ドジでノロマなカメといわれていた私に，ついにエンディングまで行ったアクションゲームができたのです。いま，俺は猛烈に感動している！　うおおぉぉぉぉ！（で）

▶ふだんから，ATMの「現金ヲ，オ忘レナクオ持チ帰リクダサイ」という声をうざったいなーと思っていたけれど，先日ついに現金をおろしたまま取るのを忘れて帰ってきてしまった。どうせうざったいなら，もっと利用者が気をつけるような声のかけ方を考えるべきだ……って，私がいうと責任逃れですか。そうですか。はは。（春麗の浦）

▶先月に引き続き就職の話だったりするが，私はいまだに職が決まっていない。ついに8月1日の解禁日を過ぎてしまった。ちなみに私は，この“解禁日”という言葉が嫌いである。「おいらは鮎じゃねぇぞぉ」と，声を大にしていいたい。ああ，入社試験で会ったあのかわいい女の子と，悪友たちと出かけた平泉の青い空をもう一度見てみたいなあ。（毛）

▶面白いもの，新しい刺激になるものを求めていたら，コンピュータに辿りついた人もいる。コンピュータという概念の面白さにとりつかれて辿りついた人もいる。生産性を上げる道具として，仕事を進めるために不可欠な道具としてしか捉えていない人は辿りつけない。そんな世界が流れている。まだ，観光地化されていない景観地の川の上流だ。（K）

▶1988年10月号の編集後記の続き（？）。行き場を失っていた2MバイトのRAMボードが3年の歳月を経てXVIのスロットに落ち着いた。これでXVIもメモリ10Mバイトだ。ところで，コプロセッサをXVIに内蔵するなど2台のX68000の間でボードの再配分を行ったら，今度は拡張I/Oボックスが不要になってしまった。世の中うまく行かないなあ。（KO）

▶今月はとにかく慌ただしかったなあ。誕生日，旅行，引っ越し，締め切りが一時期に集中したものだから体はボロボロ，全身筋肉痛でしばらく動けない状態になってしまった。編集室には某氏が突然買った某ゲームの基板まであるし……。苦しかったけど振り返ってみると結構いいこともあった。何年ぶりだろうなあ，こんな充実した夏は。（J）

▶去年は誕生日の夜に徹夜した。今年は誕生日の朝に徹夜明けした。ふだんは徹夜をあまりしないほうなのにだ。なぜかこんな日にかぎってこうなってしまう。もちろんすべて自分の責任であるから，しょうがないといえばしょうがない。しかし，なんとかしたかったなあ。うーん。忙しいとか，大変だとかいうのはきらいだけど，今月は大変だった。（A）

▶結局何処へも行けなかった。胃の出血もあばらのヒビも治らないまま。そして輝ける夏は遠ざかっていく……あ～ん。それはそうと，あたしの主夫が欲しい宣言に対し，そんな恐ろしいことやるヤツいるのかとの文書がきた。みんなぁ，誤解しちゃ困るぜよ。あたしだって昔は「なりたいもの，およめさん」だったんだから。可愛いもんだろ，え!?（E.O.）

▶基板とコンパネを買ってくる。餅は餅屋。「お願いね」と（八）君に渡す。コンパネに電源を入れてコントロールボックス代わりができあがり。最近X68000のクロックアップがはやっているようだが，自分でハッチ当てもできない人は手を出すべきじゃない。いくつか悲惨なはがきもきている。いずれにせよ，他人にすすめるものではあるまい。（U）

▶わけありでWindowsの雑誌を手がけることとなった（石を投げないで！）。誌名はTHE WINDOWSで，11月創刊を目指して準備を進めている。それにしてもIBMコンパチのDOS/Vマシンは安い。メモリ8MにHDを120M載んだ486マシンが50万円で買えてしまう。HDは80Mでいいよっていったら「お客さん1万円しか違いませんよ」だって……。（T）

## microOdyssey

ソフトが売れない，と，いう。X68000で年間に売れるソフトの本数が60万本だとして，それを100～150タイトルで割れば推し量ることができる(電脳倶楽部は除外)。ソフトハウスとパソコンショップがあんまりもうかっていないのは間違いないだろう。

対処するには買い支えしかない。X68000のゲームが減ってきているそうだが，Oh!Xで大きめに扱われるソフトだけでも月に5，6本はあるから，みんなだいたい毎月3万円分くらい(少ないか？)ソフトを買えばパソコンショップやソフトハウスの人も納得してくれるだろうか。

状況を探ろう。昨年発売されたゲームのジャンルをざっと調べてみるとシミュレーションが26本，美少女もの20本，アクション/シューティングが18本，RPG14本，アドベンチャー9本，パズル6本などとなっている。

なんとシミュレーションがいちばん多いという意外な結果である。都合上，フライトシミュレータなども含まれているが，供給過剰の感は拭えない。今年，AIIIがパッケージでの発売を見合わせたのもこの影響だろうか。次いで，昨年はエルフの参入をはじめ美少女ものが急成長した年でもあった。大小合わせるとタイトル数もかなり多い。これもかなり一般ゲームの市場を食っているようだ。シューティング関係は年末のナイアス，ソルフィース，イメージファイトによる壮絶なつぶしあいが記憶に新しい。

結局，売れ筋を見ると，対象の年齢層も広く，長く遊べるソフトが断然強い。

問題の一端は，どうしてこんなにぶつけてくるだといいたくなるほど，同じジャンルのソフトが一時期に集まることにもある。ゴールデンウィークと年末にソフトが固まるのも問題だ。今年の春もパロ，オーガスタ，ファランクス，AIIIといった超大型ソフトが続出した。結果として，「これは」と思わせるような突出感を持ったソフトが見られなくなっている。全体にゲームのレベルは向上しつつも，以前ほどの迫力は感じられない。ゲームを取り巻く環境も変化しているのだ。

ユーザーが増えれば比例してソフトの売り上げが伸びるという考え方は無条件に正しいのだろうか。「おみこし」をはじめ，シャープのX68000戦略は一貫して「お友達にも勧めましょう」式のものを展開している。たとえば，仲間内に3人のユーザーがいて，限られたおこづかいのなかで3人とも同じタイトルを買っている……ほど日本人はクールではない。タイトル数が増え続ければ，個々は目減りすることもありうる。そのうち一部のレーザーディスクのように「お友達にもソフトを貸してはいけません」といい始めるソフトハウスも現れることだろう。

古代中国の人なら漢字数文字で状況を説明しつつ，スマートに痛切な皮肉を浴びせることができるのだろうが，あいにく私には学がないのでだらだらとしてしまった。しかし，そろそろ5年目ともすれば，買い支えユーザーが緊縮財政モードに入っていてもしかたないという気もしないではないな。

そういえば，去年，ラグーンがヒットしてZOOMの社長が写真週刊誌に取り上げられたことがあった。その快挙をもたらした数字はわずか「1万本」だったのだが。　(U)

## 1991年11月号10月18日(金)発売

### 特集　空間彷徨型ゲーム大分析

スペシャルレビュー　スターウォーズ，ドラッケンなど
立体空間の表現法あれこれ/業務用ゲームにおける3D体験
CARDDRV用カードゲーム　ギャップ
SX-WINDOW環境セットアップ
全機種共通システム
Small-C活用講座　応用編

## バックナンバー常備店

| 地域 | 地区 | 店名 | 電話 |
|---|---|---|---|
| 東京 | 神保町 | 三省堂神田本店5F | 03(3233)3312 |
| | 〃 | 書泉ブックマートB1 | 03(3294)0011 |
| | 〃 | 書泉グランデ5F | 03(3295)0011 |
| | 秋葉原 | T-ZONE 7Fブックゾーン | 03(3257)2660 |
| | 八重洲 | 八重洲ブックセンター3F | 03(3281)1811 |
| | 新宿 | 紀伊国屋書店本店 | 03(3354)0131 |
| | 高田馬場 | 未来堂書店 | 03(3200)9185 |
| | 渋谷 | 大盛堂書店 | 03(3463)0511 |
| | 池袋 | リブロ池袋店 | 03(3981)0111 |
| | 〃 | 西武百貨店9F コンピュータ・フォーラム | 03(3981)0111 |
| 神奈川 | 横浜 | 有隣堂横浜駅西口店 | 045(311)6265 |
| | 〃 | 有隣堂ルミネ店 | 045(453)0811 |
| | 藤沢 | 有隣堂藤沢店 | 0466(26)1411 |
| 神奈川 | 厚木 | 有隣堂厚木店 | 0462(23)4111 |
| | 平塚 | 文教堂四の宮店 | 0463(54)2880 |
| 千葉 | 柏 | 新星堂カルチェ5 | 0471(64)8551 |
| | 船橋 | リブロ船橋店 | 0474(25)0111 |
| | 〃 | 芳林堂書店津田沼店 | 0474(78)3737 |
| | 千葉 | 多田屋千葉セントラルプラザ店 | 0472(24)1333 |
| 埼玉 | 川越 | 黒田書店 | 0492(25)3138 |
| | 川口 | 岩渕書店 | 0482(52)2190 |
| 茨城 | 水戸 | 川又書店駅前店 | 0292(31)0102 |
| 大阪 | 北区 | 旭屋書店本店 | 06(313)1191 |
| | 都島区 | 駸々堂京橋店 | 06(353)2413 |
| 京都 | 中京区 | オーム社書店 | 075(221)0280 |
| 愛知 | 名古屋 | 三省堂名古屋店 | 052(562)0077 |
| | 〃 | パソコンΣ上前津店 | 052(251)8334 |
| | 刈谷 | 三洋堂書店刈谷店 | 0566(24)1134 |
| 長野 | 飯田 | 平安堂飯田店 | 0265(24)4545 |
| 北海道 | 室蘭 | 室蘭工業大学生協 | 0143(44)6060 |

## 定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある「新規」「継続」のいずれかに○をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください。なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

**海外送付ご希望の方へ**

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700

10月号
■1991年10月1日発行　定価600円(本体583円)
■発行人　孫　正義
■編集人　橋本五郎
■発売元　ソフトバンク株式会社
■出版事業部　〒108 東京都港区高輪2-19-13 NS高輪ビル
Oh!X編集部　☎03(5488)1309
出版営業部　☎03(5488)1360 FAX 03(5488)1364
広告センター　☎03(3297)0181
■印　刷　凸版印刷株式会社

# 満開の電子(でんこ)ちゃん

作・え　岡村　祭

文月　涼（神奈川県）

出会いは、或る日、彼の部屋に行った時のことでした。当日、彼はブツを用意して私の来訪を待っており、最初に実演してくれたのが、何とあの著名な〝ゲボボディスケット〟だったのです。虜になった私は直ちに購読を申込み、それ以来、いけないとは思いつつも、変酋長のお言葉に目をウルウルさせながら止めることが出来ません。最近、会社に副業をやっていることがばれて、大激務のプロジェクトに飛ばされましたが、連絡一本でどこまででも追い掛けてくれる電脳倶楽部って、どうしてもどうしても、やめられまへんな。

# AVCフタバ

☎03(3253)7661

〒101 東京都千代田区外神田3-2-3☎03-3253-7661(代)

| 今すぐ もよりの電話から | 仙 台 022-264-3704 | 名古屋 052-452-3271 | 広 島 082-295-6873 |
|---|---|---|---|
| 札 幌 011-611-5104 | 新 潟 0252-75-4175 | 大 阪 06-311-3931 | 福 岡 092-481-2494 |

高価下取り、買取りいたします、お問合せ下さい。

AVC本店 AVCジャンプ AVCホップ 外堀通り 至新宿 電波ビル リバナ 日通ビル 至上野 ヤマギワ 中央通り 至神田 秋葉原駅

Welcome!
ご来店もどうぞ。

## X68000の情報のすべて!(当店はX68000の認定代理店です。お気軽にご相談下さい)

## X68000 超破格値で相談に応ず。 X68000

PERSONAL WORKSTATION **SUPER**

PERSONAL WORKSTATION **PROII**

SX-WINDOW、SCSIインターフェース標準装備。

拡張I/Oポートを4スロット搭載、拡張性と低価格が魅力。

SX-WINDOW標準装備。

- ●CZ-604C・TN(チタンブラック)…標準価格¥348,000
- ●CZ-623C・TN(チタンブラック)…標準価格¥498,000
- ●CZ-653C-BK・GY……………標準価格¥285,000
- ●CZ-663C-BK・GY……………標準価格¥395,000

お勧めディスプレイコーナー 組合せは自由、価格はお気軽にご相談下さい。

●ドットピッチ0.31mm
●TVチューナー搭載
●ステレオスピーカー搭載
●チルト台同梱
**CZ-613D**
標準価格¥135,000
**AVC特価**

●ドットピッチ0.39mm
●TVチューナー搭載
●ステレオスピーカー搭載
●チルト台同梱
**CZ-605D**
標準価格¥115,000
**AVC特価**

●ドットピッチ0.31mm
●TVチューナー無し
●チルト台同梱
**CZ-606D**
標準価格¥79,800
**AVC特価**

●0.31mmドットピッチ
●2モードオートスキャン
●ステレオスピーカ搭載
●チルト台同梱
**CZ-604D**
標準価格¥94,800
**AVC特価**

**熱転写カラープリンタ**
48ドット熱転写カラー漢字プリンタ
CZ-8PC5-BK
予約受付中
AVC特価

**カラードットプリンタ**
24ピン、カラー漢字プリンタ(80桁)
CZ-8PG1
標準価格¥130,000
AVC特価

**カラーイメージジェット**
カラーイメージジェット
IO-735X
標準価格¥248,000
AVC特価

**増設用ハードディスク**
80MB(CZ-604C内蔵用)
CZ-68H
標準価格¥160,000
AVC特価

**増設用ハードディスク**
40MB(CZ-602C、603C、652C、653C内蔵用)
CZ-64H
標準価格¥120,000
AVC特価

**SCSIボード**
CZ-6BS1
標準価格¥29,800
(ソフトウェア〈SCSIユーティリティ〉付)
AVC特価

**1MB増設RAMボード**
CZ-6BE1B
標準価格¥28,000

**2MB増設RAMボード**
CZ-6BE2B
標準価格¥79,000

**4MB増設RAMボード**
CZ-6BE4B
標準価格¥138,000

AVC特価

## X68000 NEW
PERSONAL WORKSTATION
**XVI** エクシヴィ

瞬速の16MHz

CZ-634C-TN
……………¥368,000
CZ-613D-TN
……………¥135,000
(スピーカ2個、チルトスタンド同梱)

AVC特価 価格はお電話で

●セットの組合せは自由!広告に出ていない他の機種はお問合せ下さい。

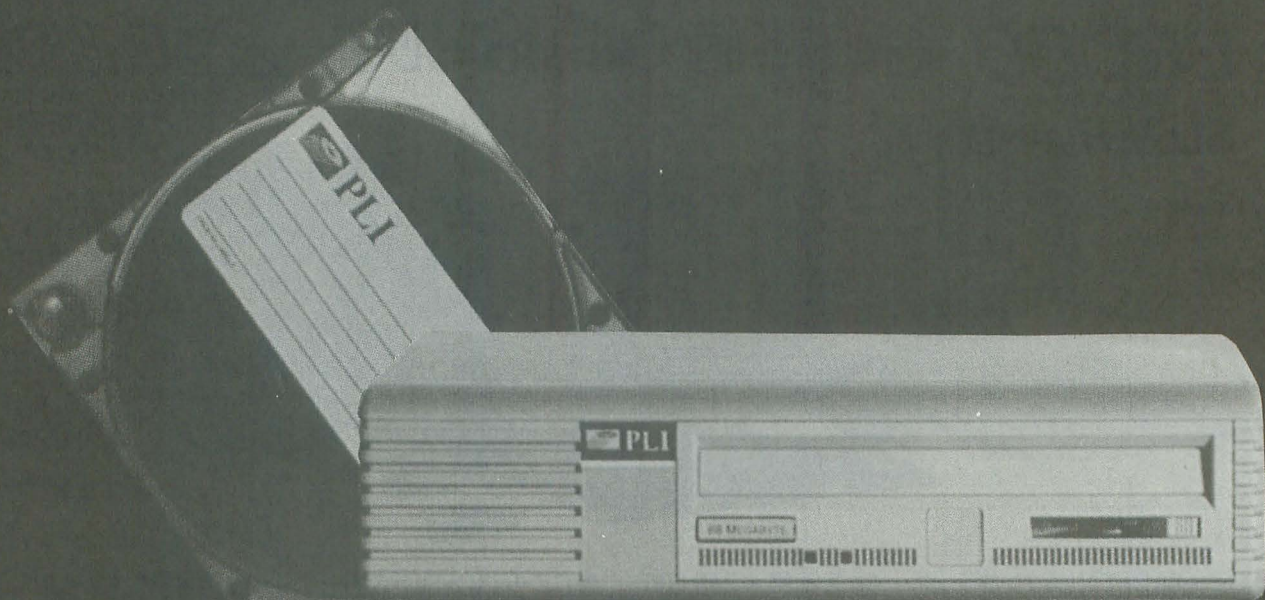
●頭金なし(手軽な電話クレジット)●製品先取り(お支払いは約1～2ヶ月後から)●低金利クレジット(1回の支払いは2,700円以上で3～48回。ボーナス併用も可)●カレッジクレジット(保証人なし。但し満20歳以上の学生の方)●18歳未満の方(ご両親が代理購入者としてお申し込み下さい)●納期(通常の場合、当社に申込書が到着後1週間以内。特に人気のある商品で品薄の場合、少々納期が遅れることがありますので御了承下さい)●完全保証(すべてメーカー保証書付。アフターケア万全)●全国代引(お届けした者に、代金をお支払いいただく方法です。但し手数料1,000円)

AM10時からPM7時まで受付 日曜・祝日も営業

☎価格は電話で値切って下さい。 ※中古も取扱っています お問合せ下さい。

# THE Removable HDD

メディア交換可能な新世代ハードディスク

# PLI Infinity 40 turbo

Oh!X 特別価格
**¥148,000**
メディア2枚サービス

CZ-6BS1
セット価格
**¥170,000**

リムーバブルハードディスクはフロッピーディスクや光磁気ディスクと同じようにメディアを抜き差しできる新世代のハードディスクです。
交換が可能になることによりデータのバックアップなどをメディアごと行なうことも可能ですし、他のInfinity 40 turboユーザーとのデータ交換なども簡単に出来るようになります。
PLI社のInfinity 40 turboはアメリカでマッキントッシュやIBM PC用として既に多大な評価を受けておりその品質は万全です。気になる平均シークタイムも20msecと固定型のハードディスクに引けを取らない高速です。BASIC HOUSEはこの大変便利なデバイスをぜひX68000ユーザーの皆さんにも使用していただきたいと思い、販売を開始いたします。

★平均シークタイム20msec
★メディア1枚当たりの容量42Mバイト
★連続耐久時間30000時間以上
★SCSIインターフェース対応
★X68000用SCSIケーブル、ターミネータ付属
★メディア1枚(40Mバイト)の価格はわずか¥18,000

※ SUPER,XVI以外の機種ではSCSIインターフェースボード(CZ-6BS1)が必要です。

**BASIC HOUSE** 計測技研

お申し込み・お問い合わせは **0286-22-9811**(代)

〒321 栃木県宇都宮市竹林町503-1 FAX 0286-25-3970

# ハードディスクを内蔵させた

超高速 12msec

# XVIがおいしい

エクシヴィー

BHオリジナル ハードディスク内蔵 **X68000 XVI 版登場!!**
CZ-634C-TN(XVI)に40M/100M/200MのSCSIハードディスクを内蔵。

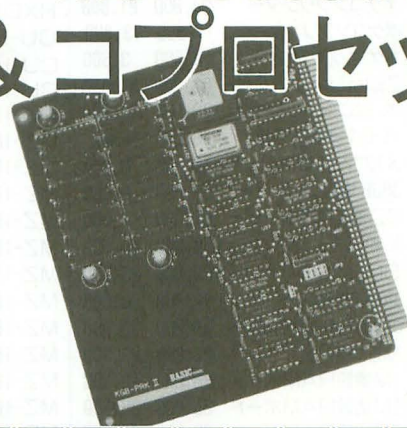
| | | |
|---|---|---|
| 40M バイト内蔵モデル **XVI40** | BH特価 | ¥ 378,000 |
| 100M バイト内蔵モデル **XVI100** | BH特価 | ¥ 428,000 |
| 200M バイト内蔵モデル **XVI200** | BH特価 | ¥ 528,000 |

**通信販売のみ!一般販売店では扱っておりません。**

※表示価格はハードディスクを内蔵させた本体のみの価格です。
※ディスプレイなどは別にお求め下さい。
※使用しているドライブの関係上、立ちあげには電源投入後約15秒で一度リセットをする必要があります。

MAX8Mバイト　MC－68881RC16

# 1枚2役・増設RAM&コプロセッサ

# KGB-X68PRKII

- ●8M増設メモリと数値演算プロセッサが1枚のボードに収まります。
- ●従来品(KGB-X68PRK)に比べて大幅なコストダウン。
- ●メモリ容量2M/4M/6M/8Mの4種類、それぞれに数値演算プロセッサ有無のモデルを用意しました。
- ●数値演算プロセッサ無しモデルではMC68881RC16の購入で簡単にグレードアップが可能です。
- ●当然、2M/4M/6Mモデルでは購入後も8Mまでのメモリ増設が可能です。

※XVIではCZ-6BE2Bとの共存ができません。購入前に弊社までご相談ください。

| | | |
|---|---|---|
| 2Mメモリ数値演算プロセッサ無し | PRKII-02 | ¥ 55,000 |
| 4Mメモリ数値演算プロセッサ無し | PRKII-04 | ¥ 90,000 |
| 6Mメモリ数値演算プロセッサ無し | PRKII-06 | ¥ 125,000 |
| 8Mメモリ数値演算プロセッサ無し | PRKII-08 | ¥ 160,000 |
| 2Mメモリ数値演算プロセッサ付属 | PRKII-12 | ¥ 85,000 |
| 4Mメモリ数値演算プロセッサ付属 | PRKII-14 | ¥ 120,000 |
| 6Mメモリ数値演算プロセッサ付属 | PRKII-16 | ¥ 155,000 |
| 8Mメモリ数値演算プロセッサ付属 | PRKII-18 | ¥ 190,000 |


**BASIC HOUSE** 計測技研
お申し込み・お問い合わせは **0286-22-9811**(代)
〒321 栃木県宇都宮市竹林町503-1 FAX0286-25-3970

**AIBIT**
アイビット電子株式会社

# SHARP
## パソコン本体から周辺機器まで品数取り揃え
# 大特価セール実施中!!

| 型名 | 品名 | 正価 | 特価 |
|---|---|---|---|
| PC-E550 | ポケコン64K | 32,000 | **25,600** |
| PC-E500BL | ポケコン | 28,800 | **19,500** |
| PC-1600K | ポケコン | 69,800 | **49,800** |
| PC-1360K | ポケコン | 36,800 | **32,800** |
| PC-1360 | ポケコン | 29,800 | **19,800** |
| PC-1262 | ポケコン | 24,800 | **19,600** |
| PC-1248DB | ポケコン | 11,000 | **9,800** |
| PC-1280 | ポケコン | 24,800 | **19,600** |
| CE-T800 | ポケコンRS-232Cコンバーター | 12,800 | **11,800** |
| CE-2H16M(16K) | | 16,000 | **11,000** |
| CE-2H32M(32K) | | 32,000 | **16,000** |
| CE-2H64M(64K) | | 45,000 | **30,000** |
| CE-212M(8K) | | 18,000 | **6,000** |
| CE-203M | ポケコンRAM32K | 32,000 | **7,000** |
| CE-202M | ポケコンRAM16K | 35,000 | **6,000** |
| CE-201M | ポケコンRAM 8K | 18,000 | **3,000** |
| CE-1600M | ポケコンRAM32K | 32,000 | **16,000** |
| CE-1600F | ポケコンフロッピードライブ | 39,800 | **34,800** |
| CE-1600P | ポケコンプリンター | 69,800 | **59,800** |
| CE-1650F | ポケコンDISK | 9,800 | **8,800** |
| CE-161 | ポケコンRAM16K | 50,000 | **3,800** |
| CE-1601M | ポケコンRAM64K | 45,000 | **30,000** |
| CE-1600E | ポケコンディスクインターフェイス | 19,800 | **17,800** |
| CE-158 | ポケコンレベルコンバター | 39,800 | **31,300** |
| CE-159 | ポケコンRAM 8K | 35,000 | **4,200** |
| CE-140T | ポケコンRS-232Cコンバター | 9,800 | **8,800** |
| CE-140F | ポケコンフロッピディスク | 49,800 | **44,800** |
| CE-130T | ポケコンRS232Cレベルコンバータ | 17,800 | **15,800** |
| CE-135T | ポケコンRS422レベルコンバータ | 9,800 | **8,800** |
| CE-123P | ポケコンプリンター | 19,800 | **17,800** |
| CE-120P | ポケコンプリンター | 24,800 | **21,800** |
| CE-126P | ポケコンプリンター | 17,800 | **13,800** |
| CE-124 | ポケコンカセットインター | 4,500 | **3,600** |
| Z-VISIONplus | Z80シュミレータ デバッカー | 59,800 | **51,000** |
| UX-1 | ホームコピーファクス | 78,000 | **69,800** |
| PA-9500 | ハイパー電子手帳 | 48,000 | **特価** |
| CZ-300F | X13"マイクロフロッピー | 79,800 | **6,000** |
| CZ-31F | 300F増設フロッピー | 59,800 | **6,000** |
| CZ-82F | CZ-802C増設フロッピー | 59,800 | **6,000** |
| CZ-501H | X1増設用ハードディスクユニット | 258,000 | **60,000** |
| CZ-6BS1 | SCSIボード | 29,800 | **23,800** |
| CZ-6BP1 | 数値演算ボード | 79,800 | **63,800** |
| CZ-6BU1 | ユニバーサルI/Oボード | 39,800 | **33,800** |
| CZ-6BM1 | MIDIボード | 29,800 | **23,800** |
| CZ-6BEIB | 1M増設RAMボード | 28,000 | **19,500** |
| CZ-6BE1 | 1M増設RAMボード | 35,000 | **29,500** |
| CZ-6BE2 | 2M増設RAMボード | 79,800 | **63,800** |
| CZ-6BE4 | 4M増設RAMボード | 138,000 | **110,400** |
| CZ-6BN1 | スキャナーボード | 29,800 | **25,300** |
| CZ-6BF1 | RS-232C増設ボード | 49,800 | **42,300** |
| CZ-6SD1 | システムラック | 44,800 | **38,000** |
| CZ-6TU | RRGBシステムチューナー | 33,100 | **26,500** |
| CZ-6BG1 | X68000GPIBボード | 59,800 | **50,000** |
| CZ-6BC1 | X68000FAXボード | 79,800 | **67,800** |
| CZ-6PV1 | ビデオプリンター | 198,000 | **158,000** |
| CZ-6BV1 | ビデオボード | 21,000 | **16,800** |
| CZ-822C | X1G MODEL30 | 118,000 | **39,800** |
| CZ-820C | X1G MODEL10 | 69,800 | **16,800** |
| CZ-128SF | X1CP/M | 9,800 | **8,500** |
| CZ-130SF | X1torbo漢字CP/M | 14,800 | **12,500** |
| CZ-115LF | X1FORTRAN | 13,800 | **11,700** |

| 型名 | 品名 | 正価 | 特価 |
|---|---|---|---|
| CZ-116LF | X1C | 13,800 | **11,700** |
| CZ-8BGR2 | グラフィックボードX1 | 14,800 | **3,000** |
| CZ-8BF1 | FDインターフェイス | 14,800 | **11,500** |
| CZ-8BK2 | X1 漢字ROM | 19,800 | **16,800** |
| CZ-8BM2 | 232Cマウスボード | 19,800 | **16,800** |
| CZ-8BE2 | 320K外部メモリー | 29,800 | **25,300** |
| CZ-8BR1 | 立体映像セット | 29,800 | **25,300** |
| CZ-8BV2 | カラーイメージボート | 39,800 | **32,000** |
| CZ-8BO1 | FDインターフェイス | 14,800 | **8,000** |
| CZ-8TM1 | X1ソフト付モデムユニット | 29,800 | **5,000** |
| CZ-8TM2 | X1ソフト付モデムユニット | 49,800 | **39,800** |
| CZ-8EB3 | 拡張i/o box | 33,800 | **28,000** |
| CZ-8LM1 | 232cケーブル | 7,200 | **6,000** |
| CZ-8LM2 | 232cクロスケーブル | 7,200 | **6,000** |
| CZ-8NJ1 | ジョイカード | 1,700 | **1,360** |
| CZ-8NT1 | トラックボール | 13,800 | **11,500** |
| CZ-8PK10 | 24ドット136桁漢字プリンター | 99,800 | **大特価** |
| CZ-8PK7 | 24ドット80桁漢字プリンター | 122,000 | **59,800** |
| CZ-8PC5BK | 48ドット熱転写カラー漢字プリンター | 96,800 | **特価** |
| CZ-8BS1 | X1FM音源ボード | 23,800 | **19,500** |
| CZ-8BK4 | X1第2水準ROM | ——— | **5,700** |
| CZ-8NJ2 | インテリジェントコントローラー | 23,800 | **18,500** |
| CZ-8NS1 | カラーイメージスキャナー | 188,000 | **149,000** |
| CZ-612DGY | 15インチ0.31TV付 | 119,800 | **80,000** |
| CZ-888C | X1ターボZIII | 169,800 | **115,000** |
| AN-S100 | アンプ付スピーカー | 36,600 | **29,500** |
| AN-X68 | キーボードシリコンカバー | 3,500 | **2,800** |
| AN-X68PRO | キーボードシリコンカバー | 3,500 | **2,800** |
| AN-1506 | ディスプレイ15P→ 変換ケーブル | ——— | **1,600** |
| HXD040 | アイティム40Mハードディスク(ITM) | 118,000 | **75,000** |
| HXD140 | 40Mハードディスク内蔵用(ITM) | 98,000 | **75,000** |
| CU-14FD | カラーディスプレーアナログ0.31 | 74,800 | **49,800** |
| CU-14KD | アナログ0.28 14"CRT | 89,800 | **59,800** |
| CU-14TV | デジタル/アナログ0.31 14"CRT TV付 | 98,800 | **64,800** |
| MZ-1D10 | 12"モノクロディスプレー | 41,800 | **25,000** |
| MZ-1D17 | 15"CRT mz-5500/6500/2 | 124,000 | **59,800** |
| MZ-1E08 | プリンターI/F 2000/2200/80B | 9,000 | **8,000** |
| MZ-1E11 | MZ-6500用 SFD I/F | 38,000 | **25,000** |
| MZ-1E64 | MZ-2000 プリンターI/F | 10,000 | **6,000** |
| MZ-1E21 | MZ-5500 GP I/F | 36,000 | **12,000** |
| MZ-1E18 | MZ2000QD用インターフェイス | 9,800 | **3,000** |
| MZ-1E33 | MZ6500パラレルI/F | 34,800 | **28,000** |
| MZ-1E45 | MZ6500 232C I/F | 50,000 | **15,000** |
| MZ-1E32 | MZ2500 パラレル I/F | 30,000 | **27,000** |
| MZ-1E44 | MZ-6500 S-RN I/F | 50,000 | **15,000** |
| MZ-1E22 | MZ-5500 GPIB I/F | 72,800 | **25,000** |
| MZ-1E29 | RS-232Cインターフェイス 300BT | 17,800 | **9,800** |
| MZ-1E01 | MZ-3500 232Cボード | 28,000 | **13,000** |
| MZ-1E14 | MZ1500 QD用インターフェイス | 9,800 | **3,000** |
| MZ-1M01 | MZ-2000/2200 16ビットボード | 78,000 | **8,000** |
| MZ-1M09 | MZ-6500 8082-2演算プロセッサ | 82,000 | **30,000** |
| MZ-1M03 | MZ-5500 数値演算 | 69,000 | **38,500** |
| MZ-1M12 | MZ-2861 8087 演算プロセッサ | 90,000 | **45,000** |
| MZ-80P4B | 136桁ドットプリンター | ——— | **48,000** |
| MZ-1P06 | ドットプリンター | 234,000 | **45,000** |
| MZ-1P27 | 水平漢字プリンター | 268,000 | **98,000** |
| MZ-1P28 | ドットプリンター漢字80桁 | 148,000 | **118,400** |
| MZ-1P10A | 24ドットプリンター漢字80桁 | 245,000 | **79,000** |
| MZ-1P22 | 熱転写漢字プリンター | 59,800 | **19,800** |
| MZ-1P29 | 漢字プリンター136桁 | 168,000 | **134,400** |
| MZ-1P30 | 136桁プリンター | 228,000 | **120,000** |

| 型名 | 品名 | 正価 | 特価 |
|---|---|---|---|
| MZ-1R01 | MZ-2000/2200Gボード | 39,800 | **10,000** |
| MZ-1R10 | MZ-5500 漢字ROM付 | 30,000 | **9,800** |
| MZ-1R09 | MZ-5500 V.RAM | 35,000 | **15,000** |
| MZ-1R06 | MZ-5500 増設RAM | 45,000 | **8,000** |
| MZ-1R12 | MZ-80B/2000/1500/700 RAM | 35,000 | **8,000** |
| MZ-1R11 | MZ-5500 256KRAM | 80,000 | **35,000** |
| MZ-1R36 | MZ-28611M増設RAM | 45,000 | **15,000** |
| MZ-1R35 | MZ-28611M増設RAM | 55,000 | **19,000** |
| MZ-1R14 | MZ-5500 辞書ROM | 40,000 | **22,000** |
| MZ-1R16 | MZ-5500 128KRAM | 30,000 | **8,000** |
| MZ-1R21 | 漢字ROM | 38,000 | **13,000** |
| MZ-1R24 | MZ-1500 辞書ROM | 22,000 | **6,000** |
| MZ-1R32 | MZ-6500RAM | 80,000 | **40,000** |
| MZ-1R28A | MZ-2500 辞書ROM | 13,000 | **10,000** |
| MZ-1S13 | MZ-1D17チルトスタンド | 12,000 | **5,000** |
| MZ-1T02 | MZ-2200 データーレコーダー | 19,800 | **8,500** |
| MZ-1T03 | MZ-5500 データーレコーダー | 12,000 | **8,500** |
| MZ-1V01 | パソコン FAX | 278,000 | **85,000** |
| MZ-1X22 | モデムユニット | 21,800 | **13,000** |
| MZ-2Z016 | MZ-5500 附属 | ——— | **5,000** |
| MZ-2Z023 | MZ-5500 GWBASIC | 50,000 | **30,000** |
| MZ-2Z029 | MZ-6500 TODAY | 68,000 | **20,000** |
| MZ-2Z064 | MZ-6500 書院RAM付 | 69,800 | **28,000** |
| MZ-2Z065 | MZ-6500 書院RAMなし | 49,800 | **15,000** |
| MZ-2Z012 | MZ-5500 附属 | ——— | **5,000** |
| MZ-2Z013 | MZ-5500 MSDOS | 25,000 | **20,000** |
| MZ-4Z001 | MZ-5500 IBM変換 | 30,000 | **8,000** |
| MZ-5521 | 本体 | 388,000 | **55,000** |
| MZ-5511 | 本体 | 288,000 | **35,000** |
| MZ-5Z013 | MZ-1500 QD通信ソフト | ——— | **3,500** |
| MZ-6F03 | ブランク QD DISK | 450 | **400** |
| MZ-6P18 | MZ-1P18、28カットシートフィーダー | 60,000 | **35,000** |
| MZ-6P29 | MZ-1P29 カットシートフィーダー | 50,000 | **37,500** |
| MZ-6P27 | MZ-1P27 カットシートフィーダー | 58,000 | **39,800** |
| MZ-6P06 | MZ-1P06 トラクターフィード | 15,000 | **7,500** |
| MZ-6P20 | MZ-1P22/17ロールホルダー | 3,100 | **2,700** |
| MZ-6Z22 | MZ-6500(50)CP/M86BASIC-3 | 10,000 | **6,000** |
| MZ-6Z25 | M-50 ストリーマユーティリティZプロセッサ | 39,800 | **15,000** |
| MZ-80T20A | MZ-80 マシンランゲージ | 6,000 | **5,000** |
| MZ-80TUB | MZ-80 バックアップ | 20,000 | **8,000** |
| MZ-80TU | MZ-80 システムプログラム | 20,000 | **8,000** |
| MZ-80T40A | MZ-80 PASCAL | 10,000 | **5,000** |
| MZ-80T70A | MZ-80 FDOS | 20,000 | **7,000** |
| MZ-8BGK | MZ-80 BGRAM2 | 39,000 | **10,000** |
| MZ-8B104 | MZ200/2200 GP IBインターフェイス | 45,000 | **18,000** |
| MZ-8BC01 | MZ200/2200 GP IBケーブル | 18,000 | **8,000** |
| MZ-1D27 | デジタル/アナログ/0.31 14"CRT TV付 MZ25/28シリーズ用 | 99,800 | **69,800** |
| MZ-1X30 | モデムホーン | 98,000 | **19,800** |
| MZ-1R26代品 | MZ-2500増設RAM | | **10,000** |
| MZ-1U09代品 | MZ-2500拡張ポート | | **7,200** |
| MZ-8376A | AXノート286N | 398,000 | **298,000** |
| IO-735X | カラープリンター | 248,000 | **180,000** |
| BF-68PRO | フィルター | 19,800 | **16,800** |
| X68000キーボード延長ケーブル(1.5m) | | 2,500 | **2,000** |
| ディスプレーケーブルアナログ15P(3m) | | 5,000 | **4,000** |
| ディスプレーケーブルアナログ15P(1.5m) | | 4,300 | **3,500** |
| ▶XC-100P | イメージプレゼンテーター | 398,000 | **298,000** |
| ▶CU-SX1 | パソコン用プロジェクター | 980,000 | **800,000** |
| ▶XC-10SC1 | 24K→15Kスキャンコンバーター | 300,000 | **240,000** |
| ▶LEDかんばん | 15cm×15cm×8文字3色 | 950,000 | **特価** |

▶印は御注文いただいてから発送まで少々時間がかかります。

**ポケコン関係周辺機器サプライ製品及シャープ関係のソフトウエア全種取扱います。**

**FM TOWNS/FM NOTE/東芝ダイナブック、周辺機器も取扱っております。**

ALBIT
アイビット電子株式会社

# X68000 XVI フェアー!!

## 大特価セール実施中!

## XVIからEXPERTまで在庫あり。即納!

'91年10月15日迄

X68000下取りセール実施中!

## X68000

オリジナルX68000セットロゴシール(大・小)とソフト2本プレゼント(10/15まで)

**XVI**
CZ-634CTN
+CZ-603D
**¥320,000**

**XVI HD**
CZ-644CTN
+CZ-603D
**¥430,000**

**EXPERT**
CZ-602CGY
+CZ-603D ¥248,000
+CZ-612D ¥268,000

**EXPERT+HD**
CZ-602CGY+HXD140
+CZ-603DGY ¥315,000
+CZ-612DGY ¥335,000
+CZ-613DGY ¥355,000

**XVI**
CZ-634CTN
+CZ-612D
**¥340,000**

**XVI HD**
CZ-644CTN
+CZ-607DTN
アイビット特価

**EXPERTII**
CZ-603C
+CZ-606D ¥270,000
+CZ-607D ¥285,000
+CZ-614D ¥310,000

**Super HD**
CZ-623CTN
+CZ-603D ¥355,000
+CZ-612D ¥375,000

**XVI**
CZ-634CTN
+CZ-607DTN
アイビット特価

**XVI HD**
CZ-644CTN
+CZ-612D
**¥450,000**

**Super**
CZ-604CTN
+CZ-603D ¥280,000
+CZ-612D ¥300,000

**PROII**
CZ-603CBK
+CZ-606DBK
**¥228,000**

## 他のディスプレイ組合せも色々あります。

ドットマトリクス漢字プリンタ(136桁)
CZ-8PK10
標準価格 ¥97,800
特価

カラーイメージジェットプリンタ
IO-735X-B
標準価格 ¥248,000
特価 **¥198,500**

電子手帳データーを自由にカッティング
MC-300
定価 ¥580,000
資料請求して下さい。

X68000 3.5インチフロッピーディスクユニット
OS-9/X68000用デバイス、ディスクプリンタ付HUMAN68K動作可
X6835-2F
標準価格 ¥80,000
特価

24ピン漢字プリンタ(80桁)
CZ-8PK7
標準価格 ¥122,000
特価 **¥59,800**

カラーイメージスキャナ
232Cケーブル,スキャナツールソフト付
JX-220X 標準価格¥168,000
特価 **¥134,500**

カラー漢字熱転写プリンター
MZ-IP22
(MZシリーズ、X1、X68000 3シリーズ使用可)
標準価格 ¥59,800
特価 **¥19,800**

HXD 040 23ms X68000
外付けハードディスク
標準価格¥118,000 特価**¥75,000**
HXD 140 X68000 内蔵用
40Mハードディスク
標準価格¥98,000 特価**¥75,000**
HXD 140は602C, 603C, 652C, 653Cの内蔵用

## X68000ソフト

| | | 正価 | 特価 |
|---|---|---|---|
| ICONEDITER | 計測技研 | ¥ 4,800 | ¥ 4,100 |
| C言語ライブラリー(X68000) | 計測技研 | ¥ 6,800 | ¥ 5,800 |
| BASIC拡張関数パッケージ | 計測技研 | ¥ 9,800 | ¥ 8,500 |
| DISK CACHER | 計測技研 | ¥ 6,800 | ¥ 5,800 |
| たーみのる2 | SPS | ¥17,800 | ¥14,250 |
| X1エミュレータ | ACCESS | ¥ 9,800 | ¥ 8,330 |
| EM 68K(エミュレータ) | ニューウェーブ | ¥30,000 | ¥25,500 |
| CP/M 68K | ニューウェーブ | ¥110,000 | ¥93,500 |
| サイクロン Ver1.2 | アンス | ¥58,000 | ¥49,300 |
| SUPER DIVICE MONITOR "T" | ブルースカイ | ¥15,000 | ¥12,000 |
| CANVAS | CZ-249GS | ¥29,800 | ¥23,850 |
| XBAS to C CHECKER | CZ-260LS | ¥ 9,800 | ¥ 7,850 |
| Multiword | CZ-225BS | ¥32,000 | ¥25,600 |
| Human68K Ver2.0 | CZ-244SS | ¥ 9,800 | ¥ 7,850 |
| C compiler Ver2.0 | CZ-245LS | ¥44,800 | ¥35,850 |
| SOUND PRO68K | CZ-214BS | ¥15,800 | ¥12,650 |
| Sampling PRO68K | CZ-215MS | ¥17,800 | ¥14,250 |
| MUSIC PRO68K | CZ-213MS | ¥18,800 | ¥15,000 |
| MUSIC Studio PRO68K Ver2.0 | CZ-261MS | ¥28,800 | ¥23,000 |
| MUSIC PRO68K MIDI | CZ-249MS | ¥28,800 | ¥23,000 |
| CARD PRO68K Ver2.0 | CZ-253BS | ¥29,800 | ¥23,850 |
| CARD活用フォーム集1 | CZ-242BS | ¥ 9,800 | ¥ 7,850 |
| SX-WINDOW Ver1.1 | CZ-278SS | ¥ 9,800 | ¥ 7,850 |
| OS-9 | CZ-219SS | ¥29,800 | ¥23,850 |
| Easypaint SX-68K | CZ-263GW | ¥12,800 | ¥10,250 |
| Teleportion PRO68K | CZ-258BS | ¥22,800 | ¥18,250 |
| Cプロフェッショナルパッケージ | マイクロウェア | ¥58,000 | ¥49,300 |

※富士通、NEC、シャープ周辺機器(拡張機器全機種、プリンター他)も常時取り扱っております。〈全商品新品完全保証付〉シャープ、カシオポケコン全機種取扱い。カタログ、価格表ご請求には、72円を添えてお願い致します。

通信販売のお問い合せ、御注文は
**0426-45-3001(本店)**
FAX.0426-44-6002
●営業時間/10:00~19:00●電話受付/20:00迄可●定休日/水曜日
SHARP SUPER XEX SHOP
アイビット電子株式会社 〒192 東京都八王子市北野町560-5

上記の広告商品はすべて店頭販売もしております。

**全国通信販売 北海道から沖縄まで**

★送料はご注文の際にお問い合わせ下さい。
★掲載の商品は、すべて新品、保証書付きです。
★掲載の商品は充分用意してありますが、ご注文の際は、在庫の確認の上、現金書留または、銀行振込でお申し込み下さい。全商品クレジットでも扱っております。
★お申し込みの際は必ず電話番号を明記して下さい。
★商品、品切れの節はご容赦下さい。

富士銀行八王子支店 (普)1752505

●本誌発売時には上記価格よりさらにお求めやすい価格に変更されている場合があります。●この広告の商品にはすべて送料・消費税は含まれておりません。

# X68000 本体はもちろん周辺機器も充実！

## X68000 XVI セット
CZ-634C＋CZ-606D
ハードディスク内蔵モデルも特価にて販売いたします。

標準価格合計 447,800円
㊙特価／お電話にて

## X68000 SUPER セット
CZ-604C＋CZ-606D
ハードディスク内蔵モデルも特価にて販売いたします。

標準価格合計 467,800円
¥308,000

## X68000 PROⅡ セット
CZ-653＋CZ-606D
ハードディスク内蔵モデルも特価にて販売いたします。

標準価格合計 364,800円
¥278,000

直接ご来店頂けない場合は、通信販売もご利用いただけます。お電話でお申し込みください。
☎(052)332-5688

**銀行振込**
各店舗に御予約、ご注文いただきましたら、最寄の銀行から当社指定銀行口座へ『電信扱』にてお振り込み下さい。手数料はお客様負担になります。

**代金引き替え配送**
お電話で商品の注文が出来ます。お客様宅へ配達時、商品と引き替えにお代金をお支払いいただきます。商品代金の他に手数料がかかります。

**クレジット**
お電話にてお申込みいただきましたら折り返し弊社より専用申込用紙をお送りいたします。
必要事項記入の上ご返送下さい。

いずれも商品在庫をご確認の上お申し込みください。

広告に掲載されていない商品も全店特価にて取り扱っております。もちろん全品メーカー保証付です。クレジットでの購入も可能です。(3回から60回まで)
お電話お待ちしております。

※表示価格には消費税は含まれておりません。

新型ディスプレイモニタ **CZ-614D**

標準価格 135,000円
OA特価

新型ディスプレイモニタ **CZ-607D**

標準価格 99,800円
OA特価

14″ディスプレイモニタ **CZ-606D**

標準価格 79,800円
OA特価

カラープリンター **CZ-8PC5** (ブラック グレー)

標準価格 96,800円
¥78,000

新型カラーイメージスキャナー **JX-220X**

標準価格 168,000円
OA特価

TVディスプレイモニタ **CZ-613D**

標準価格 135,000円
¥99,800

TVディスプレイモニタ **CZ-605D**

標準価格 115,000円
¥79,800

21型ディスプレイモニタ **CU-21HD**

標準価格 148,000円
OA特価

カラーインクジェットプリンター **IO-735X** (ブラック グレー)

標準価格 248,000円
¥198,000

光磁気ディスク **CZ-6MO1** カートリッジ付

標準価格 480,000円
¥384,000

アイテック社製ハードディスク **TX-80** (ブラック・グレー)
X68000専用 80MB SASI,SCSI両対応
標準価格 108,000円
¥88,000

アイテック社製ハードディスク **TX-130** (ブラック・グレー)

X68000専用 130MB SCSI方式
標準価格 138,000円
¥108,000

アイテック社製ハードディスク **TX-180** (ブラック・グレー)

X68000専用 180MB SCSI方式
標準価格 185,000円
¥148,000

今、「IO-735X」をお買い上げの方へ！
「バナナプリント」
デモ版を
プレゼント致します。

X68000のグラフィック(PIC形式等)をA3版までのフルカラーでプリント可能！

MIDI音源ユニット＋インターフェースボードセット
ローランド ¥69,000 **CM-32L**
システムサコム ¥19,800 **SX68M**
標準価格合計 88,800円
¥74,000

MIDI音源ユニット＋インターフェースボードセット
ローランド ¥129,000 **CM-64**
システムサコム ¥19,800 **SX68M**
標準価格合計 148,800円
¥125,000

I・Oデータ機器製 純正互換増設RAMボード

| 型番 | 内容 | 価格 |
|---|---|---|
| PIO-6BE1A | (1MB内蔵増設RAMボード) | ¥17,800 |
| PIO-6BE2-2M | (2MB増設RAMボード) | ¥35,800 |
| PIO-6BE4-4M | (4MB増設RAMボード) | ¥61,800 |

HAL研究所
ファインスキャナー256
(数量限定！)
X68000専用ハンディイメージスキャナー
読み取り幅105mm
解像度100/200dpi
30% OFF

標準価格 39,800円
¥28,000

SHARP純正 拡張インターフェースボード
※実装方法などは各店の「PRO STUFF」までお気軽にご相談ください!!

| 型番 | 標準価格 | 価格 |
|---|---|---|
| CZ-6BE1 | 標準価格 38,000円→ | OA特価 |
| CZ-6BE1B | 標準価格 28,000円→ | OA特価 |
| CZ-6BP1 | 標準価格 79,800円→ | OA特価 |
| CZ-6BS1 | 標準価格 29,800円→ | OA特価 |
| CZ-6BF1 | 標準価格 49,800円→ | OA特価 |
| CZ-6BM1 | 標準価格 28,000円→ | OA特価 |
| CZ-6EB1 | 標準価格 88,000円→ | OA特価 |
| CZ-6BV1 | 標準価格 21,000円→ | OA特価 |
| CZ-6BN1 | 標準価格 29,800円→ | OA特価 |

SHARP純正 拡張インターフェースボード
XVIシリーズ専用タイプ
CZ-6BE2A
(XVI専用内蔵2MB増設RAMボード)
59,800円→ OA特価
CZ-6BE2B
(CZ6BE2A増設用 2MBRAM)
54,800円→ OA特価
CZ-6BP2
(XVI専用内蔵数値演算プロセッサー)
45,800円→ OA特価

| 店舗 | 電話 | 店舗 | 電話 |
|---|---|---|---|
| 札幌店 | 011-210-8812 | 大須店 | 052-265-1650 |
| 仙台店 | 022-268-5541 | 京都店 | 075-344-0347 |
| 東京店 | 03-3255-9188 | 大阪店 | 06-632-4233 |
| 横浜店 | 045-314-6634 | 大阪日本橋店 | 06-646-3169 |
| 浜松店 | 053-458-3755 | 岡山店 | 0862-21-4133 |
| 名古屋店 | 052-332-5233 | 広島店 | 082-240-9669 |
| 名古屋アメ横店 | 052-264-9715 | 福岡店 | 092-714-0030 |
| アメ横2F店 | 052-262-6909 | 福岡ユーテク店 | 092-733-8931 |

X68000 PROSHOP
㈱OAシステムプラザ
本社 愛知県名古屋市中区大井町3-
OA

# R&Rメディア

## （特価セット）

| | | |
|---|---|---|
| CZ-623C-TN | 定価 | ¥498,000 |
| CZ-606D-TN | 定価 | ¥ 79,800 |
| 合　計 | | ¥577,800 |
| R&R提供価格 | | ¥378,000 |

| | | |
|---|---|---|
| CZ-604C-TN | 定価 | ¥348,000 |
| CZ-606D-TN | 定価 | ¥ 79,800 |
| 合　計 | | ¥427,800 |
| R&R提供価格 | | ¥298,000 |

## XVI（お買い得セット）

| | | |
|---|---|---|
| CZ-634C-TN | 定価 | ¥368,000 |
| CZ-606D-TN | 定価 | ¥ 79,800 |
| 合　計 | | ¥447,800 |
| R&R提供価格 | | ¥3?0,000 |

## （ミュージックコンピュータセット）

| | | |
|---|---|---|
| CZ-634C-TN | 定価 | ¥368,000 |
| CZ-606D-TN | 定価 | ¥ 79,800 |
| SX-68M | 定価 | ¥ 19,800 |
| CM-32L | 定価 | ¥ 69,000 |
| MA-12C(2台) | 定価 | ¥ 28,000 |
| MUSIC STUDIO PRO-68K V2.0 | 定価 | ¥ 28,000 |
| 合　計 | | ¥592,600 |
| R&R提供価格 | | ¥450,000 |

## （グラフィックコンピュータセット）

| | | |
|---|---|---|
| CZ-634C-TN | 定価 | ¥368,000 |
| CZ-606D-TN | 定価 | ¥ 79,800 |
| CZ-8PC5 | 定価 | ¥ 96,800 |
| Z's STAFF PRO-68K V2.0 | 定価 | ¥ 58,000 |
| 合　計 | | ¥602,600 |
| R&R提供価格 | | ¥460,000 |

ただ今、X68000のセットをお買い上げ頂きますと
「V'BALL」「熱血高校サッカー編」「ダウンタウン熱血物語」
のいずれか1本をプレゼント／

| | 定価 | R&R提供価格 |
|---|---|---|
| **▼プリンタ（ケーブル付）** | | |
| CZ-8PC5 | ¥ 96,800 | ¥ 77,000 |
| CZ-8PG1 | ¥130,000 | ¥104,000 |
| IO-735XB | ¥248,000 | ¥173,000 |
| **▼増設メモリ** | | |
| CZ-6BE1 | ¥ 35,000 | ¥ 27,800 |
| PIO-6BE1A | ¥ 25,000 | ¥ 19,800 |
| PIO-6BE2 | ¥ 50,000 | ¥ 39,800 |
| PIO-6BE4 | ¥ 88,000 | ¥ 69,800 |
| CZ-6BE1B | ¥ 28,000 | ¥ 21,800 |
| CZ-6BE2A | ¥ 59,800 | ¥ 47,800 |
| **▼その他のオプション** | | |
| CZ-6BS1 | ¥ 29,800 | ¥ 23,800 |
| CZ-6VT1 | ¥ 69,800 | ¥ 56,000 |
| CZ-6BM1 | ¥ 26,800 | ¥ 21,500 |
| CZ-6BV1 | ¥ 21,000 | ¥ 16,900 |
| CZ-8NS1 | ¥188,000 | ¥150,000 |
| **▼ソフトウェア** | | |
| SX-WINDOW V1.1 | ¥ 9,800 | ¥ 7,840 |
| Easy Paint SX-68K | ¥ 12,800 | ¥ 10,240 |
| Multiword PRO-68K | ¥ 32,000 | ¥ 25,600 |
| Teleportion PRO-68K | ¥ 22,800 | ¥ 18,240 |

## コンピュータミュージック

| | | |
|---|---|---|
| CM-32L | 定価 | ¥ 69,000 |
| SX-68M | 定価 | ¥ 19,800 |
| MUSIC STUDIO PRO-68K V2.0 | 定価 | ¥ 28,000 |
| 合　計 | | ¥116,800 |
| R&R提供価格 | | ¥ 97,000 |

| | | |
|---|---|---|
| CM-64 | 定価 | ¥129,000 |
| SX-68M | 定価 | ¥ 19,800 |
| MUSIC STUDIO PRO-68K V2.0 | 定価 | ¥ 28,000 |
| 合　計 | | ¥176,800 |
| R&R提供価格 | | ¥149,000 |

| | | |
|---|---|---|
| SC-55 | 定価 | ¥ 69,000 |
| SX-68M | 定価 | ¥ 19,800 |
| MUSIC STUDIO PRO-68K V2.0 | 定価 | ¥ 28,000 |
| 合　計 | | ¥116,800 |
| R&R提供価格 | | ¥ 97,800 |

| ▼ゲームソフト | 定価 | R&R提供価格 |
|---|---|---|
| 生中継68 | ¥ 9,800 | ¥ 7,840 |
| スターモビール | ¥ 7,200 | ¥ 5,760 |
| イース | ¥ 9,600 | ¥ 8,160 |
| サブナック | ¥ 7,800 | ¥ 6,240 |
| フューチャー・ウォーズ | ¥ 9,800 | ¥ 7,840 |
| ループス | ¥ 7,800 | ¥ 6,240 |
| エアーコンバットII | ¥ 8,800 | ¥ 7,040 |
| アークスオデッセイ | ¥ 8,800 | ¥ 7,040 |
| ソルフィース | ¥ 8,800 | ¥ 7,040 |
| 遥かなるオーガスタ | ¥12,800 | ¥10,240 |
| エイトレイクスゴルフクラブ | ¥ 5,800 | ¥ 4,640 |
| 上海II | ¥ 6,800 | ¥ 5,440 |
| たーみのる2 | ¥17,800 | ¥14,240 |
| 銀河英雄伝説II | ¥ 9,800 | ¥ 7,840 |
| 銀河英雄伝説II DX+Kit | ¥ 5,000 | ¥ 4,000 |
| ファランクス | ¥ 8,800 | ¥ 7,040 |
| 天下統一 | ¥ 9,800 | ¥ 7,840 |
| ランペルール | ¥ 9,800 | ¥ 7,840 |
| 三国志II | ¥14,800 | ¥11,840 |
| 提督の決断 | ¥14,800 | ¥11,840 |
| ロードス島戦記 | ¥ 9,800 | ¥ 7,840 |
| 麻雀悟空 天竺への道 | ¥ 9,800 | ¥ 7,840 |
| 大戦略III'90 | ¥ 9,800 | ¥ 7,840 |
| ドラッケン | ¥ 9,700 | ¥ 7,760 |
| インペリアルフォース | ¥ 8,800 | ¥ 7,040 |
| コラムス | ¥ 7,800 | ¥ 6,240 |
| ダーウィンズ・ジレンマ | ¥ 7,800 | ¥ 6,240 |
| 殺しのドレス3 | ¥ 7,800 | ¥ 6,240 |
| ブリッツクリーク | ¥ 9,800 | ¥ 7,840 |
| アルシャーク(特製Tシャツ付) | ¥ 9,800 | ¥ 7,840 |

★未発売ソフトの予約も承ります★

※掲載商品の価格は、全て消費税別です。

**正社員を募集しています。**
パソコンが好きで、夢と野心にあふれた人がいいですね。……お待ちしています。

●取扱い商品 NEC・富士通・エプソン・シャープ（メーカー保証付） ソフト、各種サプライ用品

商品は、電話またはFAX（お客様の電話番号をお忘れなく）でご注文下さい。
お支払は銀行振込でお願いいたします。入金確認後の発送となります。 ローンも可。
送料・消費税は別途いただきますので、ご了承下さい。
掲載分以外の商品も扱っておりますので、ご相談下さい。
込先：富士銀行 西大井支店（普）1358191 アール・アンド・アール・メディア㈱


アール・アンド・アール・メディア株式会社
140 東京都品川区西大井6-10-10
TEL.03-3777-7335
FAX.03-3777-6448

価格に自信あり!!

## OAB特選〜X68000シリーズセット　★本体・ディスプレイセットでお買い上げの方にはゲームソフト2本付

**①X68000XVI**
- CZ-634C-TN
- CZ-614D-TN
- MD-2HD 20枚

定価合計￥503,000
特価
￥TEL下さい!!

**②X68000XVI-HD**
- CZ-644C-TN
- CZ-614D-TN
- MD-2HD 20枚

定価合計￥653,000
特価
￥TEL下さい!!

**③X68000 PROII**
- CZ-653C-BK/GY
- CZ-605D-BK/GY
- MD-2HD 20枚

定価合計￥400,000

●SX-WINDOW搭載!!

安くて表示できません。

**④X68000PRO II-HD**
- CZ-663C-BK/GY
- CZ-605D-BK/GY
- MD-2HD 20枚

定価合計￥510,000

安くて表示できません。

☆本体、モニターのみの方は、さらにお安くなります。

★X68000XVI お買い上げの方に＋￥10,000にて販売致します。!!

| ①CZ-6BMI | ②CZ-8NJ2 | ③BF-68PRO | ④MB68881RC16 |
|---|---|---|---|
| MIDIボード | サイバースティック | CRTフィルタ | 数値演算プロセッサー |
| 定価￥26,800 | 定価￥23,800 | 定価￥19,800 | CZ-6BP2と同等 |

**X68000 SUPER-HD**
- SX-WINDOW搭載
- SCSI I/F 装備
- 80MBハードディスク搭載
- 3MB大容量メモリ装備
- 高解像度グラフィック

**⑤X68000 SUPER-HD**
- CZ-623C-TN(チタン)
- CZ-614D-TN(チタン)
- MD-2HD 20枚

定価合計￥633,000
特価￥415,000

### X68000 特選OABセット ★本体のみ単品OK.!!

① CZ-604C-TN(新品)＋CZ-606D-TN(新品)　3セット限り………特価￥298,000
② CZ-604C-TN(新品)＋CZ-614D-TN(新品)　1セット限り………特価安くて表示できません。
③ CZ-603C-BK(新品)＋CZ-603D-BK(新同品)　3セット限り………特価￥218,000
④ CZ-662C-GY(新品)＋CZ-606D-GY(新同品)　2セット限り………特価￥228,000

# オーエーブレイン　全国通販

●オフコンからパソコンまで
幅広〜い品揃え。おまかせあれ!! お電話くださいネ!
☎03-5688-3621

★全商品保証書付。専門のアドバイザーがお客様のニーズに親切に対応します。
★初期不良・輸送トラブル等に迅速に対応し、即交換させていただきます。
★送料は、着払いとなります。

●ご注文、お問合せは…毎日午前10時から午後8時まで
●下取・買取は電話で見積りしております。責任を持って下取りさせて頂き
●商品のお届けは…入金確認後、即日発送致します。

## 周辺機器コーナー

### プリンターセットコーナー

- CZ-6PVI(カラービデオプリンター) 定価￥198,000……▶特価￥147,800
- CZ-8PC3(24ドット熱転写カラープリンター) 定価￥65,800……▶特価￥52,800
- CZ-8PK10(24ピン漢字ドットプリンター・136桁) 定価￥97,800……▶特価￥70,800
- CZ-8PG1(24ピンカラー漢字ドットプリンター・80桁) 定価￥130,000……▶特価￥91,800
- CZ-8PG2(24ピンカラー漢字ドットプリンター・136桁) 定価￥160,000……▶特価￥113,800
- IO-735XB(カラーイメージェットプリンター) 定価￥248,000……▶特価￥169,000

### X68000用ソフトウェアー・コーナー

| | 定価 | 特価 |
|---|---|---|
| ①CZ-212BS(BUSINESS) | ￥68,000 | ￥53,000 |
| ②CZ-220BS(DATA) | ￥58,000 | ￥45,000 |
| ③CZ-215MS(Sampling) | ￥17,800 | ￥13,800 |
| ④CZ-221HS(NEW Print Shop) | ￥10,800 | ￥15,500 |
| ⑤CZ-227BS(TOP財務会計) | ￥200,000 | ￥158,000 |
| ⑥CZ-226BS(CARD) | ￥229,800 | ￥23,000 |
| ⑦CZ-223CS(Communicabion) | ￥19,800 | ￥115,500 |
| ⑧CZ-213MS(MUSIC) | ￥18,800 | ￥14,800 |
| ⑨CZ-211LS(C compiler) | ￥39,800 | ￥31,000 |
| ⑩C-TRACE(キャスト) | ￥68,000 | ￥52,000 |
| ⑪EW(イースト) | ￥38,000 | ￥29,000 |

### 特選品

■ CZ-8PC5(48ドット熱転写カラー漢字プリンター)(定価￥96,800)安くて表示できません。

## X68000用周辺機器コーナー

| 型番 | 品名 | 定価 | 特価 |
|---|---|---|---|
| CZ-6BE1 | IBM増設RAMボード | ￥35,000 | ￥25,200 |
| CZ-6BE1B | IBM増設RAMボード | ￥28,000 | ￥20,200 |
| CZ-6BE2 | 2MB増設RAMボード | ￥79,800 | ￥58,700 |
| CZ-6BE4 | 4MB増設RAMボード | ￥138,000 | ￥102,200 |
| CZ-6BF1 | 増設用RS-232Cボード | ￥49,800 | ￥36,700 |
| CZ-6BG1 | GP-IBボード | ￥59,800 | ￥43,200 |
| CZ-6BM1 | MIDIボード | ￥26,800 | ￥19,200 |
| CZ-6BN1 | スキャナ用パラレルボード | ￥29,800 | ￥21,700 |
| CZ-6BP1 | 数値演算プロセッサボード | ￥79,800 | ￥58,700 |
| CZ-6BO1 | ユニバーサルI/Oボード | ￥39,800 | ￥29,200 |
| CZ-6EB1/BK | 拡張I/Oボックス | ￥88,000 | ￥63,700 |
| CZ-6VT1/BK | カラー・イメージ・ユニット | ￥69,800 | ￥50,700 |
| CZ-8NM24 | マウス | ￥6,800 | ￥4,700 |
| CZ-8NT1 | マウストラックボール | ￥9,800 | ￥6,700 |
| CZ-8NS1 | カラーイメージスキャナ | ￥188,000 | ￥134,700 |
| CZ-6BC1 | FAXボード | ￥79,800 | ￥58,700 |
| CZ-8TM2 | モデムユニット | ￥49,800 | ￥36,700 |
| CZ-64H | 増設ハードディスク | ￥120,000 | ￥86,700 |
| CZ-6TU GY/BK | RGBシステムチューナー | ￥33,100 | ￥23,700 |
| BF-68PRO | 高性能CRTフィルタ | ￥19,800 | ￥14,700 |
| CZ-6MO1 | 光磁気ディスクユニット | ￥450,000 | ￥326,700 |
| CZ-6BS1 | SCSIインターフェースボード | ￥29,800 | ￥21,700 |
| CZ-6BL2 | LANボード | ￥298,000 | ￥216,700 |

## I・O DATA 増設RAMボード　限定

| ●1MB増設PAMボード | ●2MB増設RAMボード | ●4MB増設RAMボード |
|---|---|---|
| PIO-6BE1-A | PIO-6BE2-2M | PIO-6BE4-4M |
| 定価￥25,000 | 定価￥50,000 | 定価￥88,000 |
| 特価￥16,800 | 特価￥33,300 | 特価￥58,300 |

## ハードディスク　★その他特価品有!TEL下さい!!

- ■シャープ CZ-64H……特価￥86,000
- CZ-68H……特価￥118,000
- ■ロジテック LHD-200…特価￥218,000
- ■アイテム HXD-040…特価￥88,000
- HXD-042…特価￥95,000
- ■アイテック 価格応談 ●TX-80…特価￥77,800
- ●TX-130…特価￥97,800
- ●TX-180…特価￥130,000
- ★SCSIボード………特価￥22,000

## オーエーブレイン今月の特価品!! 台数限定 お早目に

| | 定価 | 特価 |
|---|---|---|
| KGB-X68PRKII-02 | ￥55,000 | ￥42,800 |
| PRKII-04 | ￥90,000 | ￥70,200 |
| PRKII-06 | ￥125,000 | ￥97,500 |
| PRKII-08 | ￥160,000 | ￥124,800 |
| PRKII-12 | ￥85,000 | ￥66,300 |
| PRKII-14 | ￥120,000 | ￥93,600 |
| PRKII-16 | ￥155,000 | ￥121,000 |
| PRKII-18 | ￥190,000 | ￥148,000 |
| MC-6888 IRC | ￥38,000 | ￥28,500 |

- 〈開発ツール〉●C-コンパイラPRO68KV.2 定価￥44,800 CZ-245IS……特価￥33,000
- 〈C言語〉●C&Professional Pack 定価￥58,000……特価￥40,500
- 〈データベース〉●CARD PRO68K Ver.2.0 定価￥29,800 CZ-253BS……特価￥21,000
- 〈音楽〉●Music studio PRO68K Ver.2.0 定価￥28,800 CZ-261MS……特価￥21,300
- 〈CGシール〉●CANVAS PRO68K 定価￥29,800 CZ-249GS……特価￥22,200
- 〈通信〉●Tlepotion PRO68K 定価￥22,800 CZ-258BS……特価￥
- 〈ワープロ〉●Multiword PRO68K 定価￥32,000 CZ-225BS……特価￥
- 〈グラフィック〉●Z's STAFF PRO68K (シャフト)定価￥58,000……特価￥
- 〈グラフィック〉●C-TRACE68 Ve 定価￥98,000……特価￥

## 中古パソコン

| | |
|---|---|
| PC-9801RA2 | ￥248,000より |
| PC-9801RX2 | ￥180,000より |
| PC-9801VX21 | ￥175,000より |
| PC-9801VM21 | ￥140,000より |
| PC-9801VM2 | ￥125,000より |
| PC-9801F2 | ￥48,000より |
| PC-9801EX2 | ￥180,000より |
| PC-9801UV21 | ￥115,000より |
| PC-9801LV21 | ￥143,000より |
| PC-286V | ￥125,000より |
| PC-286VE | ￥130,000より |
| PC-286L | ￥110,000より |
| PC-286LS | ￥220,000より |
| PC-8801FH | ￥48,000より |
| PC-8801MA | ￥55,000より |
| X68000 | ￥140,000より |
| X68000(HD) | ￥190,000より |
| X1ターボZII | ￥58,000より |
| FM77AV40EX | ￥45,000より |
| 200ラインCRT | ￥8,000より |
| 400ラインCRT | ￥30,000より |
| 80桁プリンタ | ￥15,000より |
| 135桁プリンタ | ￥35,000より |

●その他多数有り、お問い合せ下さい。

## ユニット

- FD-1155D(5インチ)……￥9
- FD-1155C(5インチ)……￥8
- FD-1165A(8インチ)……￥3
- FD-1137D(3.5インチ)……￥9
- D-5146H(5インチ40MB)…￥29
- D-3142(3.5インチ40MB)…￥29
- D-3148(3.5インチSISC)…￥30
- 外付8インチ2ドライブ……￥20
- 外付5インチ2ドライブ……￥30

## 通信販売によるご購入方法(お電話でお申し込み下さい。)

**現金一括払い**
銀行振込：電信扱いにてお振込下さい。手数料はお客様負担となります。
現金書留：住所、氏名、電話番号、商品名、使用機種、メディア等をお書き添えのうえ、現金書留にて当社までお送り下さい。

**クレジット**
専用のお申し込み用紙をお送り致しますので、必要事項をご記入・捺印のうえ、ご返送下さい。
※未成年者の方は、保護者のご承認を受けてからお申し込み下さい。

**振込先**
●第一勧業銀行　御徒町支店　(普)1376679　オーエーブレイン
●朝日信用金庫　本店　(普)334833　オーエーブレイン

★クレジットは1〜60回払いで月々5,000円よりご自由に設定できます。

オーエーブレイン　〒110 東京都台東区台東1-28-4　TEL & FAX 5688-3621

■流通事情により、広告表示よりお安くなる場合もございます。まずは、お電話下さい。■ビジネス・ゲームセットもございます。

ACCESS

# X1エミュレータ

好評発売中

定価¥9,800

X1エミュレータはX68000上でX1シリーズのアプリケーションを実行するためのソフトエミュレータです。X1のアプリケーションを完全にソフトウェアのみでエミュレートしているため、X1上での実行速度と比較して、平均3〜5倍程度おそくなりますが、X68000のマシン上に実現した仮想X1マシンを楽しめます。また、X1とX68000の相互間でファイルを転送するためのユーティリティと専用ケーブルが付属しますので、X1上で作り上げたソフトの資産をX68000上に移行することも簡単にできます。

## X1エミュレータの機能

- X1エミュレータはX1に相当する機能をエミュレート。
この仮想コンピュータには最大4つのドライブが仮想的に接続。
- X1エミュレータからみたドライブはHuman68kのドライブ上にあるファイルで仮想的に実現。このファイルはX1用の5" 2Dディスクのイメージをファイル転送ユーティリティでまるごと転送したもの。
- X1エミュレータで仮想的に実現したX1は仮想ドライブから起動。
このため仮想ドライブ用ファイルには、X1を立ち上げるために必要なHuBASICやCP/Mなどのシステムプログラムが必要。
- X1エミュレータでは、X1の持つVRAMを含むメモリイメージやZ80CPUを仮想的にソフトウェアで実現。

## ファイル転送ユーティリティ

**ディスク転送**

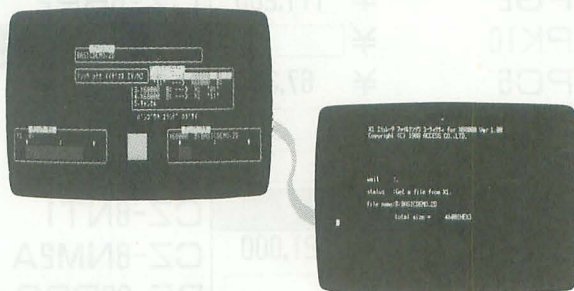
X1ディスク⟷X68000 Human68k(5" 2Dディスクイメージファイル)

- X1エミュレータではHuman68k上のディスクイメージファイルを仮想ドライブとして使用。

**ファイル転送**

X1 BASIC：CP/M⟷X68000 Human68k

- X1で作ったプログラム&データをX68000上で使用。

※付属の専用ケーブルをX1とX68000に接続してファイルを転送します。

## X1エミュレータ Q&A

**Q.** ファイル転送のために別途RS-232Cケーブルを買わないといけないのですか？
**A.** 専用のケーブルが付属しますのでその必要はありません。

**Q.** X1BASICのプログラムをX68000上のX-BASICで使えますか？
**A.** 通常のセーブではコードが違うので使用できませんが、アスキーセーブしたファイルであればX-BASIC上でそのままロード可能です。

**Q.** TurboBASICで作成した住所録などの漢字を含んだデータがあるのですがX68000上にファイル転送できますか？
**A.** X1TurboもX68000も漢字はシフトJISコードなのでファイルの転送は可能です。ただし、漢字ROMを必要とするものはサポートしていません。

**Q.** Turbo用のソフトは動きますか？
**A.** X1用のみでTurbo専用のソフトは動きません。

**Q.** ゲームは動きますか？
**A.** 純粋にBASICでかかれたものは動きますが、プロテクトがかかったものや直接ハードをアクセスするような市販のゲームは動きません。

＊タイミング等ハードウェアに依存するようなソフトは、原理上実行できない、もしくは正常に動作しない場合がありますのでご注意ください。
＊一部サポートしていない機能があります。

**X1エミュレータ通信販売** 購入希望として住所、氏名、電話番号をお知らせください。注文書をお送り致します。

＊この商品価格には消費税は含まれておりません。
＊CP/Mはデジタルリサーチ社の商標です。
文中のソフトウェアは各社の商標です。
＊製品の仕様、名称は予告なく変更する場合もございますのであらかじめご了承ください。

有限会社アクセス 〒101 東京都千代田区神田神保町1-64 神保町協和ビル7F
☎03(3233)0200(代) FAX.03(3291)7019

# パソコン/ワープロ通信ネットワークサービス J&P HOT LINE

J&P HOTLINEは
私の生きがいです。

## 岩渕 剛さん 20歳
(JH283382 Mr.CZ)

ソフトウェア開発者

取材しようと連絡をとると、「長期出張中」とのこと。それもそのはず。原子力発電所のシステム開発を手掛けている岩渕さんは、現在、東海原子力発電所にてお仕事中。休日・夜間も関係なしの交替勤務とか。20歳とは思えない落ち着いた口調で、誇りを持ってお仕事のことを話してくださいました。

## 東北ボード企画対策実行委員会迷幹事!?

岩渕さんの自宅は仙台市。会社もほんとうは(?)仙台市。というわけで地域別BBSの東北ボードで活躍中です。仕事柄、地方出張が多くおまけに深夜勤務や休日勤務などの不規則勤務のため、友達と会ってワイワイ騒ぐ機会もなかなかないとのこと。だから、HOTLINEが友達との日常的な交流の場。HOTLINEなら勤務のないときに自由にアクセスしてワイワイがやがや騒ぐこともできるからです。

東北ボードはパワフルなメンバーが多く、岩渕さんも鍛えられてパワフルライターに。100行メッセージも続々。加えて、オフラインミートは壮絶を極めるとか。地域ボードだからといって東北の人だけが集っているという訳ではなく、日本全国を網羅しているため、大規模ミートになるとのこと。その楽しさは筆舌に尽くしがたく、岩渕さんは社員旅行をけって(カイシャノヒト、ゴメンナサイ)オフラインミートに参加したこともあるそうです。その熱意を認められ、今では「東北ボード企画対策実行委員会迷幹事」という名誉ある役職の彼。まさに東北ボードは生きがい。「仕事の都合などでアクセスできない日があるとイライラしてしまいます」ということばが印象的でした。

### ★★特別企画!! 東北ボード紹介★★

本州最北端の東北でありますが、実際は半分のネットワーカーが地域外というヘンなボードです。宮城県を拠点に、東京・大阪・福岡・沖縄の宮古というとんでもない地域の方がいらっしゃっています。オフラインミートも大変盛んで、月に1回は開催しており、年に2回程度大規模なミートを展開します! そのときは全国各地から旧知のメンバーや新しいメンバーが磁石に引きよせられるがごとく集まって来ます。どんな新人さんでも1回参加すればたちまち明日からはお友達! ここはそんな人間味溢れるボードです。 (by NANNO)

### J&P HOTLINEへのご入会はスタータキットで。

お求めは、下記のお店へ。又は現金書留にて、¥3,000+¥90(消費税3%)=¥3,090を事務局までお送り下さい。
すぐにスタータキットをお送りします。

お問い合わせは 〒556 大阪市浪速区日本橋西1-6-5 上新電機株式会社
J&P HOTLINE事務局宛 TEL.(06)632-2521

### スタータキットのお求めはJ&P各店でどうぞ。

| 店名 | 住所 | 電話 |
|---|---|---|
| 渋谷店 | 東京都渋谷区道玄坂2丁目28番4号 | ☎(03)3496-4141 |
| 町田店 | 東京都町田市森野1丁目39番16号 | ☎(0427)23-1313 |
| 八王子店 | 東京都八王子市旭町1番1号八王子そごう7F | ☎(0426)26-4141 |
| 立川店 | 東京都立川市幸町4-39-1 | ☎(0425)36-4141 |
| 本厚木店 | 厚木市中町3-4-3 | ☎(0462)25-1548 |
| 富山店 | 富山市桜町2-1-10 | ☎(0764)32-3133 |
| 金沢店 | 金沢市入江2-63 | ☎(0762)91-1130 |
| 寺地店 | 金沢市寺地2-3 | ☎(0762)47-2524 |
| 大須店 | 名古屋市中区大須4丁目2-48 | ☎(052)262-1141 |
| テクノランド | 大阪市浪速区日本橋5丁目6番7号 | ☎(06) 634-1211 |
| メディアランド | 大阪市浪速区日本橋5丁目8番26号 | ☎(06) 634-1511 |
| コスモランド | 大阪市浪速区難波中2丁目1番17号 | ☎(06) 634-3111 |
| U. S. LAND | 大阪市浪速区日本橋4丁目9番15号 | ☎(06) 634-1411 |
| ビジネスランド | 大阪市北区梅田1-1-3大阪駅前第3ビルB2 | ☎(06) 348-1881 |
| 梅田店 | 大阪市北区小松原町1-10 | ☎(06) 362-1141 |
| 高槻店 | 高槻市高槻町11番16号 | ☎(0726)85-1212 |
| くずは店 | 枚方市楠葉花園町15番2号 | ☎(0720)56-8181 |
| 千里中央店 | 豊中市新千里東町1-3 SENCHU PAL 2番街4F | ☎(06) 834-4141 |
| 摂津富田店 | 高槻市大畑町24-10 | ☎(0726)93-7521 |
| 寝屋川店 | 寝屋川市緑町4-20 | ☎(0720)34-1166 |
| 藤井寺店 | 藤井寺市岡2丁目1番33号 | ☎(0729)38-2111 |
| 岸和田店 | 岸和田市土生町2451-3 | ☎(0724)37-1021 |
| さんのみや1ばん館 | 神戸市中央区八幡通3-2-16 | ☎(078)231-2111 |
| 西宮店 | 兵庫県西宮市河原町5-11 | ☎(0798)71-1171 |
| 伊丹店 | 伊丹市昆陽池1-63 | ☎(0727)77-5101 |
| 姫路店 | 姫路市東延末1丁目1番住友生命姫路南ビル1F | ☎(0792)22-1221 |
| 京都寺町店 | 京都市下京区寺町通仏光寺下ル恵比須之町549 | ☎(075)341-3571 |
| 京都近鉄店 | 京都市下京区烏丸通七条下ル東塩小路町702 | ☎(075)341-5769 |
| 和歌山店 | 和歌山市元寺町4丁目4番地 | ☎(0734)28-1441 |
| 奈良1ばん館 | 奈良市三条町478-1 | ☎(0742)27-1111 |
| 郡山インター店 | 大和郡山市横田693-1 | ☎(07435)9-2221 |
| 熊本店 | 熊本市手取本町4-12 | ☎(096)359-7800 |

T4910217910602 雑誌02179-10